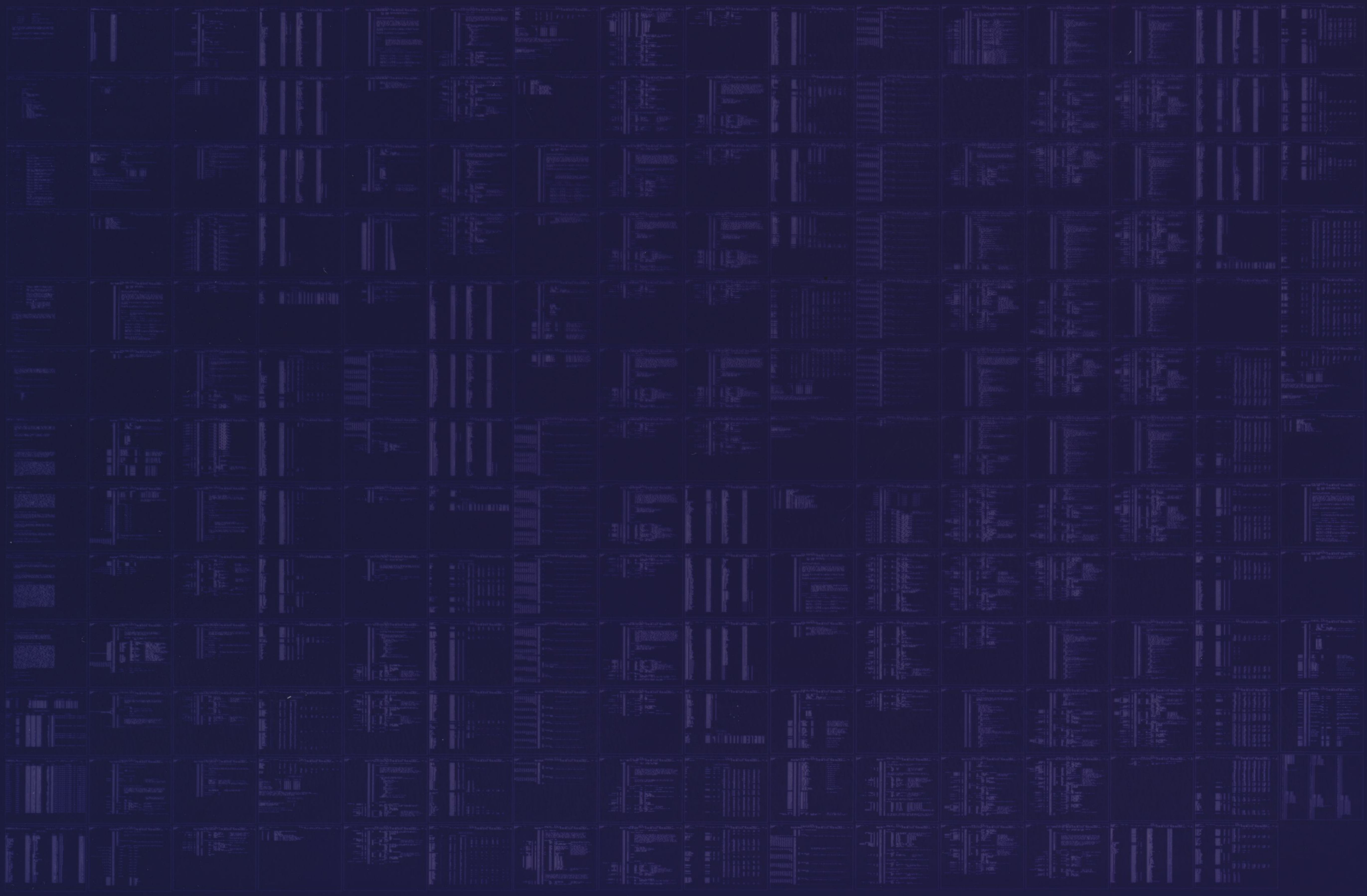
ECKAX-5.2

VAX 11/750 CPU
CLUSTER EXERCISER

EP-ECKAX-DL-5.2
1 OF 4 JUL 1986
COPYRIGHT© 1980-86

digital
MADE IN USA

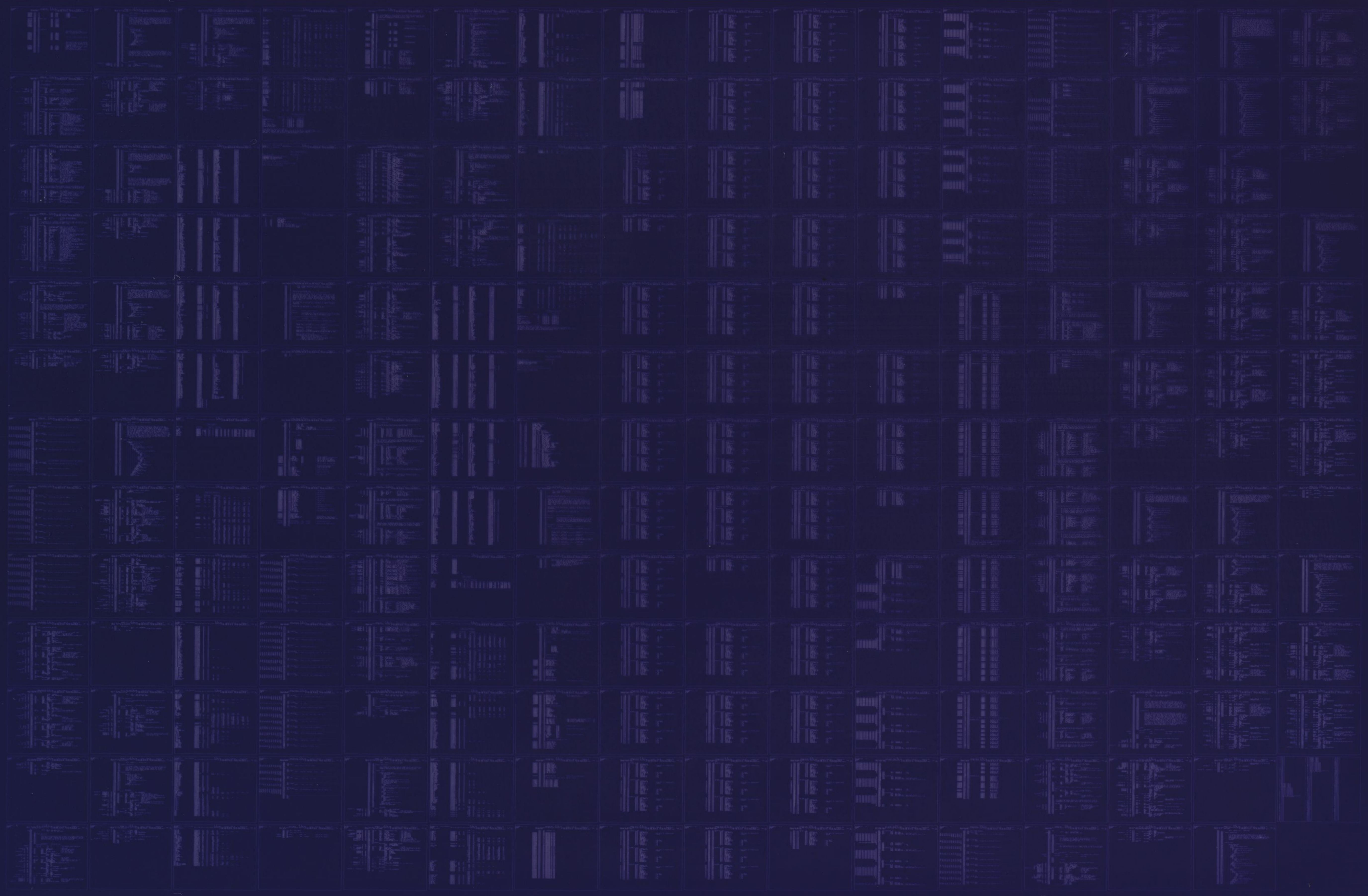


ECKAX-5.2

VAX 11/750 CPU
CLUSTER EXERCISER

EP-ECKAX-DL-5.2
2 OF 4 JUL 1986
COPYRIGHT© 1980-86

digital
MADE IN USA

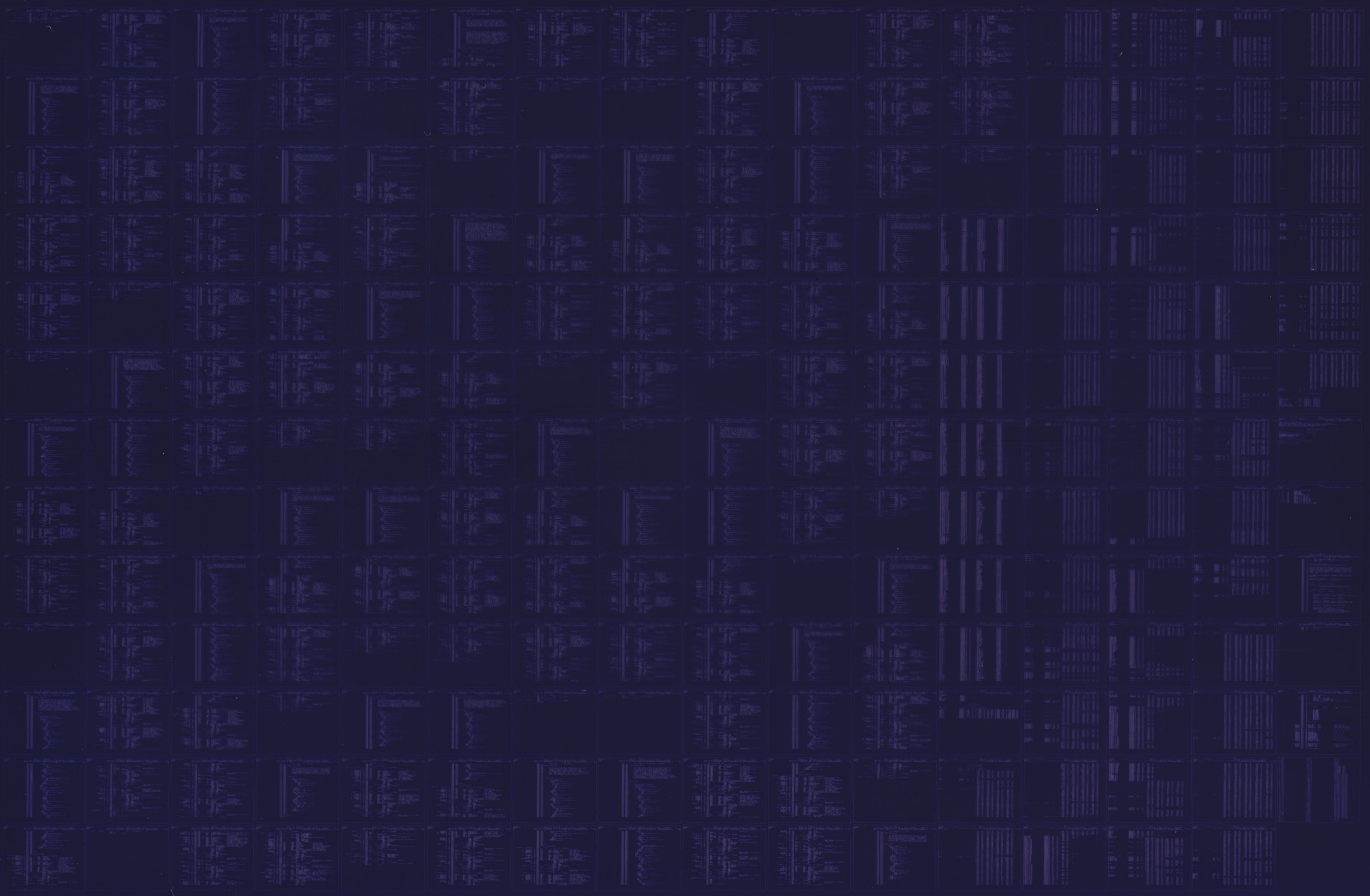


ECKAX-5.2

VAX 11/750 CPU
CLUSTER EXERCISER

EP-ECKAX-DL-5.2
3 OF 4 JUL 1986
COPYRIGHT© 1980-86

digital
MADE IN USA

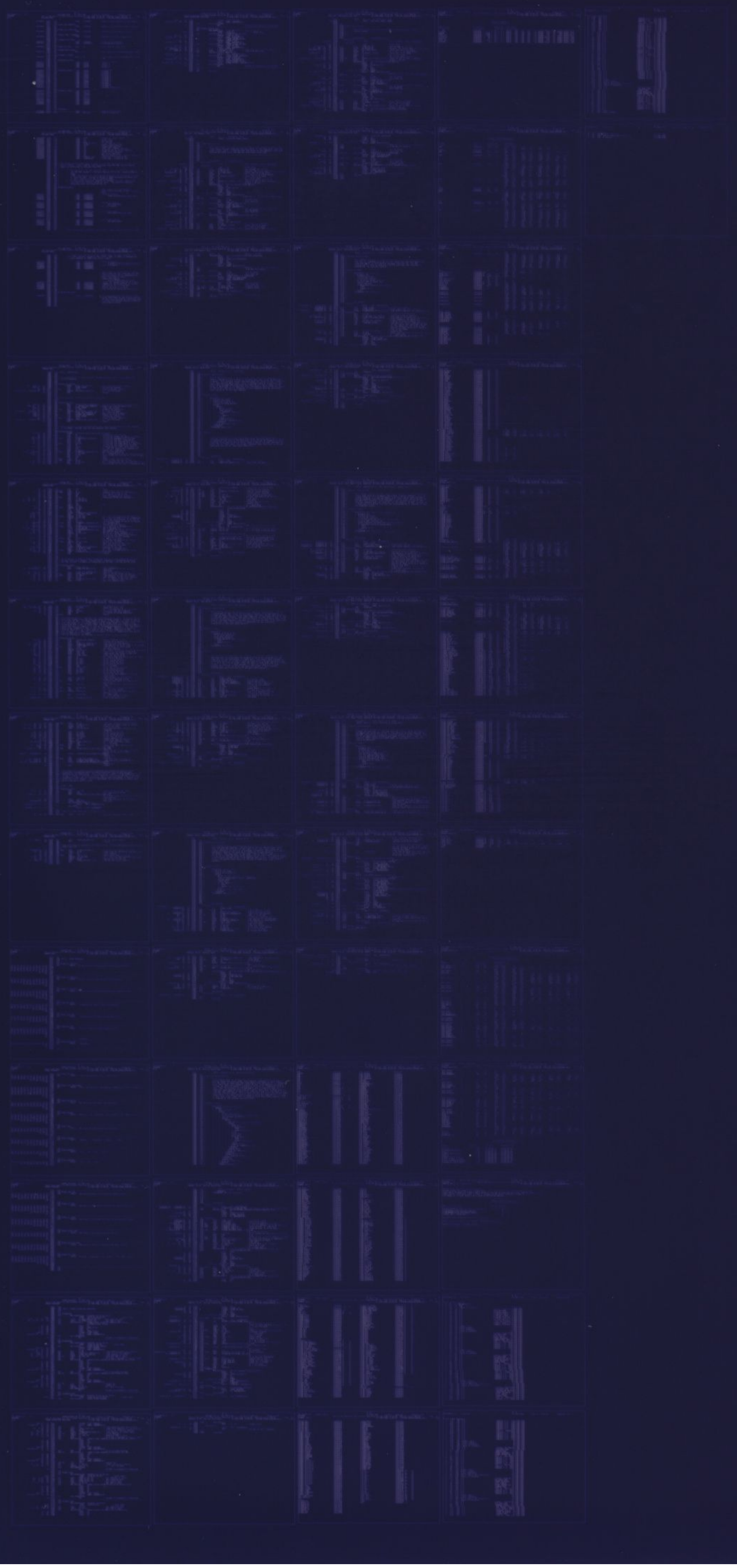


ECKAX-5.2

VAX 11/750 CPU
CLUSTER EXERCISER

EP-ECKAX-DL-5.2
4 OF 4 JUL 1986
COPYRIGHT© 1980-86

digital
MADE IN USA



IDENTIFICATION

Product Code: ZZ-ECKAX-5.2

Product Name: ECKAX 11/750 CPU CLUST EXCZR

Product Date: February 1986

Maintainer: Base Systems Diagnostic Engineering

may not be provided or otherwise made available to any other person except for use on such system and to one who agrees to these license terms. Title to and ownership of the software shall at all times remain in DEC.

The information in this software is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

DEC assumes no responsibility for the use or reliability of its software on equipment which is not supplied by DEC.

Table of Contents

1.0 MAINTENANCE HISTORY

2.0 ABSTRACT

3.0 REQUIREMENTS
  3.1 Hardware Requirements
  3.2 Software Requirements
  3.3 Prerequisites

4.0 ASSUMPTIONS

5.0 OPERATING INSTRUCTIONS

6.0 PROGRAM FUNCTIONAL DESCRIPTION

7.0 TEST DESCRIPTIONS
  7.1 Reserved Processor Registers
  7.2 Power Fail (init switch)
  7.3 Power Fail (breaker)
  7.4 Processor Halts (Arch. Defined)
  7.5 Processor halts (Arch. Undefined)
  7.6 WCS Memory
  7.7 WCS Execution
  7.8 Machine Check
  7.9 TU58 Exerciser
  7.10 TU58 Exerciser Read/Write
  7.11 TU58 Exerciser (With MRSP)
  7.12 TU58 Exerciser Read/Write (With MRSP)
  7.13 PCS Memory Test
  7.14 PCS Execution Test

8.0 B

1.0 Maintenance History

Rev	Date	Reason
1.0	6 Jun 80	Initial Release
2.0	22 Sep 80	ECKAX7 2.0 - Updated to include TU58 interrupt tests and IPL check.
		ECKAX5 2.0 - Micro instructions changed in WCS executable tests.
		ECKAX1 1.1 - FAO directive changed to ASCII code in error message.
		ECKAX1 1.1 - Header changed to reflect diagnostic revision level.
3.0	22 Oct 80	ECKAX6 2.0 - Control Store parity error subtest added. Bug in READ_GPR subroutine fixed. Loop count changed from 13 to 12.
		ECKAX7 3.0 - DS$_BREAK added befor all data checks for ^C response within 3 seconds.
		ECKAX1 1.2 - Header changed to reflect diagnostic revision level.
3.1	10 Nov 80	ECKAX5 2.1 - Debug branch removed from moving inversions subtest and $DS_BREAK added.
		ECKAX6 2.1 - Check for Cache hit added to WCS error check routine.
		ECKAX1 1.3 - Header changed to reflect diagnostic revision level.
		ECKAX1 1.4 - Header changed to reflect diagnostic revision level.
3.2	19 Feb 80	ECKAX3 2.0 - Bug fix for power fail battery backup added.
		ECKAX6 2.2 - Bug fix for don't care on MC VA in logout.
		ECKAX1 2.0 - Changed subtest 2 and 3 to be general read only and write only. IPR 3F added to write only and removed from unused list.
		ECKAX7 3.1 - Changed delay time for read dir subtest to wait longer for tape to rewind.
3.3	20 July 1981	NO CHANGE TO ANY MODULE - Correction to sections 7.2 and 7.3 of this document.

3.4	3 May 1982	ECKAX3 3.4 - Changed exit routine of Test 3 to accomodate temporarily a hardware problem.
4.0	3 Jan 1982	ECKAX 4.0 - Added new test for new PCS/WCS module (tests 13 and 14 Altered tests 5 & 6 for coverage on the enhanced WCS. (1k or 2k)
5.0	3 Mar 1983	ECKAX 5.0 - Added test 5 for coverage of Processor Halts Architecture Undefined.(Subtest in test 5 that need to use invaild SCBB are not in this test,since,a bug in VAX-11/750 MicroCode on handle Invalid SCBB was found.) Added tests 11 and 12 for coverage of new ROM with MRSP of TU58.
5.2	February 1986	Redefined IPR's to jive withthe starlet. Modified code for problem report Eckax5: Tests 5 & 6 test with cache off. Modified code for problem report Eckax6: Subtest 1 doesn't check bit 0 of the memory under test. Subtest 4 tests row 0, 1, 2 but not row 3.

2.0 Program Abstract

This program is a level 3 diagnostic that tests most portions of the VAX 11/750 CPU cluster hardware which are not specified by the VAX architecture. The areas tested by this program are reserved processor registers, power fail, processor halts, WCS, machine checks, and TU58. This program must run in a standalone environment under control of the Diagnostic Supervisor.

3.0 Requirements

3.1 Hardware:

The VAX 11/750 computer system. A WCS module if testing WCS.

3.2 Software:

Diagnostic Supervisor

3.3 Prerequisites:

ZZ-EVKAA VAX Hardcore Instruction Test.

4.0 Assumptions

Memory and the load path are good.

5.0 Operating Instructions

5.1 Loading and Starting

Refer to the "VAX - 11 Diagnostic Systems Users Guide for instructions on how to load and start the Diagnostic Supervisor and how to load and execute programs under the Diagnostic Supervisor. The operator must attach and select the CPU (KA750) before starting the program. This program will not run with memory management set on by the operator.

5.2 Program size

The program will take 266 pages (136192 bytes) of memory.

5.3 Run time

Average runtime for a fault free pass without executing manual tests is aproximately 10 minutes.

5.4 Section names

The section names are as follows:

DEFAULT
MANUAL
TU58
POWERFAIL
HALT
MCHK
WCS
IPR

5.5 Standard error message

The diagnostic supervisor produces the standard header message which contains the program ID, the test number, the subtest number, the error number (1), the class of error (HARD), the unit under test (KA0), and a program supplied message describing the function executed.

Example:

```
**** VAX 11/750 Specific CPU Cluster Exerciser - ZZ-ECKAX-3.2  ****
 PASS 1  TEST 2  SUBTEST 3  ERROR 1 1-JAN-1981  00:07:13.42
 HARD ERROR WHILE TESTING KA0:      Description of failure
```

6.0 Program Functional Description

7.0 Test Descriptions

7.1 ECKAX1: RESERVED PROCESSOR REGISTERS

This test will verify that all internal processor register addresses not implemented on the VAX-11/750 will cause a reseved operand fault for a MTPR or MFPR instruction. Internal processor registers specific to the VAX-11/750 are checked to not cause reserved operand faults depending if they are read-write, read only, or write only. This tests does not check each bit in each individual register.

7.2 ECKAX3: POWER FAIL (INIT SWITCH)

This test checks the VAX-11/750 power failure and recovery mechanism. For this test to function properly the right front panel select switch must be in the HALT/RESTART position. This test consists of a series of manual operator intervention subtests. These subtests will check that the Restart Parameter Block (RPB) is recognizable and is able to recover properly in the event of a power failure. The power failure induced by pushing the initialize switch on the front panel of the 11/750. This simulates a power fail without removing all power from the system. When this test is complete the operator will be prompted to execute the next test using the breaker switch on the bus if it is equipped with a battery backup. If a negative response is received the test is exited and the power fail test is complete. Before entering a negative responce the front panel switch must be set in the HALT position for the halt test to function properly.

7.3 ECKAX3: POWER FAIL (BREAKER)

This test consists of a series of manual operator intervention subtests. These subtests are the same as in the Power Fail (init) test but instead of pushing the init switch to simulate the power failure, the breaker at the bus of the 11-750 is thrown causing a power failure. Attempts to simulate system power failure using either the front panel key or the processor circuit breaker on the back panel removes the filtering action on switching transients by the line filter and the normal power-fail sequence may be defeated. For this test to function properly the right front panel select switch must be in the HALT/RESTART position.The operator is asked if the system under test is equipped with a battery backup. If a negative response is received the test is exited and the power fail test is complete. Before entering a negative responce the front panel switch must be in the HALT position for the other tests to function properly.

7.4 ECKAX4: PROCESSOR HALT (ARCH. DEFINED)

This test checks the various ways the VAX-11/750 processor can be halted in an architecturally defined manner. This includes executing a HALT instruction in kernel mode, attempting certain references to an invalid interrupt stack and executing change mode instructions when the IS bit of the PSL is set.

7.5 ECKAX4: PROCESSOR HALTS (ARCH.UNDEFINED)

This module checks the various ways the VAX-11/750 processor can be halted in an ARCHITECTURALLY UNDEFINED manner. These ways includes a change mode instruction with vertor service on IS and invalid vector Halt.

7.6 ECKAX5: WRITABLE CONTROL STORE

This test checks that the WCS module, if present, may be loaded as a memory and also may be executed from. WCS tested as a memory consists of data path, row select, address path and moving inversions.

7.7 ECKAX5: WRITABLE CONTROL STORE EXECUTION

The executable test consists of an entry/exit subtest, next field addressing, and an execution data path subtest.

7.8 ECKAX6: MACHINE CHECK EXCEPTIONS AND INTERRUPTS

This test checks that a reference to non-existent memory will cause a machine check exception while attempting to read and an interrupt attempting to write. A Control Store parity error machine check is also forced.

7.9 ECKAX7: TU58 EXERCISER

This test exercises the TU58 on the VAX-11/750 by performing initialization, self diagnostic and no operation. All information received from the TU58 is checked for errors. This subtest does not write or read the TU58. All operations performed deal with the controller only.

7.10 ECKAX7: TU58 EXERCISER READ/WRITE

This test exercises the TU58 on the VAX-11/750 by performing position, read (normal - flag-driven and interrupt-modes - and two versions of modified) and write (normal - flag- and interrupt-driven modes - and two versions of modified) commands. Zero padding of blocks and records are also checked for write commands. All packets are checked for errors. Timers are setup to time out if a byte transaction between the host and TU58 is not completed within a reasonable amount of time. If any errors are detected, they are reported. This test assumes the scratch tape cartridge used is in good condition. If a marginal tape is used bits may be dropped during the read modified test which reads with an increased threshold and will consequently report false data errors. Any tape may be used, but existing data will be destroyed. Before the tape is written, the first directory will be read. If the directory read is not scrat .ch , the operator will be prompted to either overwrite the tape or abort this test. If overwriting the tape is chosen, an attempt to write scrat .ch in the first directory area will be made. If an operational error occurs it will be reported, but error detection in this subtest is limited and the primary function is to give the operator an opportunity to remove a tape containing valuable data from the drive and inserting a scratch tape before it is destroyed. An IPL check is included to check that the TU58 will not interrupt the CPU unless the IPL is 16 or lower. No data is checked in the IPL check subtest.

7.11 ECKAX7: TU58 EXERCISER (With MRSP)

This test exercises the TU58 (With MRSP) on the VAX-11/750 by performing initialization, self diagnostic and no operation. All information received from the TU58 is checked for errors. All errors detected are reported. This test does not write or read the TU58. All operations performed deal with the controller only.

7.12 ECKAX7: TU58 EXERCISER READ/WRITE (With MRSP)

This test exercises the TU58 on the VAX-11/750 by performing position, read (normal - flag-driven and interrupt-modes - and two versions of modified) and write (normal - flag- and interrupt-driven modes - and two versions of modified) commands. Zero padding of blocks and records are also checked for write commands. All packets are checked for errors. Timers are setup to time out if a byte transaction between the host and TU58 is not completed within a reasonable amount of time. If any errors are detected, they are reported. This test assumes the scratch tape cartridge used is in good condition. If a marginal tape is used bits may be dropped during the read modified test which reads with an increased threshold and will consequently report false data errors. Any tape may be used, but existing data will be destroyed. Before the tape is written, the first directory will be read. If the directory read is not scrat .ch , the operator will be prompted to either overwrite the tape or abort this test. If overwriting the tape is chosen, an attempt to write scrat .ch in the first directory area will be made. If an operational error occurs it will be reported, but error detection in this subtest is limited and the primary function is to give the operator an opportunity to remove a tape containing valuable data from the drive and inserting a scratch tape before it is destroyed. An IPL check is included to check that the TU58 will not interrupt the CPU unless the IPL is 16 or lower. No data is checked in the IPL check subtest.

7.13 ECKAX8: PCS Memory Test

7.14 ECKAX8: PCS Execution Test

8.0 Bibliography

8.1 VAX - 11 Diagnostic Systems User Guide

```
                                                                          L  1
ZZ-ECKAX-5.2     Map                                                               Fiche 1  Frame L1        Sequence 11
ECKAX                                                                  3-MAR-1986 09:45        VAX-11 Linker V04-00                 Page     1

                                                  +--------------------------+
                                                  ! Object Module Synopsis !
                                                  +--------------------------+

Module Name      Ident              Bytes      File                                  Creation Date        Creator
-------- --      -----              -----      ----                                  -------------        -------
ECKAX1           5.2                 1386 [TAYLOR.ECKAX]ECKAX1.OBJ;1             3-MAR-1986 09:45   VAX/VMS Macro V04-00
ECKAX2           5.2                 1742 [TAYLOR.ECKAX]ECKAX2.OBJ;1            25-FEB-1986 09:36   VAX/VMS Macro V04-00
ECKAX3           5.2                 4447 [TAYLOR.ECKAX]ECKAX3.OBJ;1            25-FEB-1986 13:14   VAX/VMS Macro V04-00
ECKAX4           5.2                 8488 [TAYLOR.ECKAX]ECKAX4.OBJ;1            25-FEB-1986 13:15   VAX/VMS Macro V04-00
ECKAX5           5.2                 4046 [TAYLOR.ECKAX]ECKAX5.OBJ;1            21-FEB-1986 09:49   VAX/VMS Macro V04-00
ECKAX6           5.2                 3671 [TAYLOR.ECKAX]ECKAX6.OBJ;1            21-FEB-1986 09:50   VAX/VMS Macro V04-00
ECKAX7           5.2                28541 [TAYLOR.ECKAX]ECKAX7.OBJ;1            21-FEB-1986 09:51   VAX/VMS Macro V04-00
ECKAX8           5.2                 4548 [TAYLOR.ECKAX]ECKAX8.OBJ;1            21-FEB-1986 10:05   VAX/VMS Macro V04-00

                                                 +----------------------------+
                                                 ! Program Section Synopsis !
                                                 +----------------------------+

Psect Name       Module Name       Base     End               Length               Align                 Attributes
-     - -        -----------       ----     ---               ------               -----                 ----------
$HEADER                          00000200 00000291 00000092 (         146.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE,  RD,NOWRT,NOVEC
                 ECKAX1          00000200 00000291 00000092 (         146.) PAGE 9
$TSTCNT                          00000294 00000297 00000004 (           4.) LONG 2 NOPIC,USR,OVR,REL,LCL,NOSHR,NOEXE,  RD,NOWRT,NOVEC
                 ECKAX2          00000294 00000297 00000004 (           4.) LONG 2
                 ECKAX3          00000294 00000297 00000004 (           4.) LONG 2
                 ECKAX4          00000294 00000297 00000004 (           4.) LONG 2
                 ECKAX5          00000294 00000297 00000004 (           4.) LONG 2
                 ECKAX6          00000294 00000297 00000004 (           4.) LONG 2
                 ECKAX7          00000294 00000297 00000004 (           4.) LONG 2
                 ECKAX8          00000294 00000297 00000004 (           4.) LONG 2
ARGLIST                          00000298 0000060F 00000378 (         888.) LONG 2 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                 ECKAX2          00000298 000002C7 00000030 (          48.) LONG 2
                 ECKAX3          000002C8 00000327 00000060 (          96.) LONG 2
                 ECKAX4          00000328 000003B7 00000090 (         144.) LONG 2
                 ECKAX5          000003B8 000003FF 00000048 (          72.) LONG 2
                 ECKAX6          00000400 00000423 00000024 (          36.) LONG 2
                 ECKAX7          00000424 000005BB 00000198 (         408.) LONG 2
                 ECKAX8          000005BC 0000060F 00000054 (          84.) LONG 2
CLEANUP                          00000610 000006F5 000000E6 (         230.) LONG 2 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,  WRT,NOVEC
                 ECKAX1          00000610 000006F5 000000E6 (         230.) LONG 2
DATA                             00000800 0000099C 0000019D (         413.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,  WRT,NOVEC
                 ECKAX1          00000800 0000099C 0000019D (         413.) PAGE 9
DISPATCH                         000009A0 00000AEF 00000150 (         336.) LONG 2 NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE,  RD,NOWRT,NOVEC
                 ECKAX2          000009A0 000009B7 00000018 (          24.) LONG 2
                 ECKAX3          000009B8 000009E7 00000030 (          48.) LONG 2
                 ECKAX4          000009E8 00000A17 00000030 (          48.) LONG 2
                 ECKAX5          00000A18 00000A47 00000030 (          48.) LONG 2
                 ECKAX6          00000A48 00000A5F 00000018 (          24.) LONG 2
                 ECKAX7          00000A60 00000ABF 00000060 (          96.) LONG 2
                 ECKAX8          00000AC0 00000AEF 00000030 (          48.) LONG 2
DISPATCH X                       00000AF0 00000B07 00000018 (          24.) LONG 2 NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE,  RD,NOWRT,NOVEC
                 ECKAX1          00000AF0 00000B07 00000018 (          24.) LONG 2
```

```
Psect Name          Module Name          Base     End           Length             Align                 Attributes
----------          -----------          ----     ---           ------             -----                 ----------
ECKAX1                                   00000C00 00000D45 00000146 (         326.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX1               00000C00 00000D45 00000146 (         326.) PAGE 9
ECKAXA                                   00000E00 00001134 00000335 (         821.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX2               00000E00 00001134 00000335 (         821.) PAGE 9
ECKAXC                                   00001200 00001D52 00000B53 (        2899.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX3               00001200 00001D52 00000B53 (        2899.) PAGE 9
ECKAXD                                   00001D54 00002B60 00000E0D (        3597.) LONG 2 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,  WRT,NOVEC
                    ECKAX4               00001D54 00002B60 00000E0D (        3597.) LONG 2
ECKAXE                                   00002C00 000035E4 000009E5 (        2533.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX5               00002C00 000035E4 000009E5 (        2533.) PAGE 9
ECKAXF                                   00003600 00004103 00000B04 (        2820.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX6               00003600 00004103 00000B04 (        2820.) PAGE 9
ECKAXG                                   00004200 0000644C 0000224D (        8781.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX7               00004200 0000644C 0000224D (        8781.) PAGE 9
ECKAXH                                   00006600 00006FDE 000009DF (        2527.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX8               00006600 00006FDE 000009DF (        2527.) PAGE 9
INITIALIZE                               00006FE0 00007088 000000A9 (         169.) LONG 2 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,  WRT,NOVEC
                    ECKAX1               00006FE0 00007088 000000A9 (         169.) LONG 2
SUMMARY                                  0000708C 000070D9 0000004E (          78.) LONG 2 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,  WRT,NOVEC
                    ECKAX1               0000708C 000070D9 0000004E (          78.) LONG 2
TEST_001                                 00007200 0000754C 0000034D (         845.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX2               00007200 0000754C 0000034D (         845.) PAGE 9
TEST_002                                 00007600 000078AC 000002AD (         685.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX3               00007600 000078AC 000002AD (         685.) PAGE 9
TEST_003                                 00007A00 00007CCA 000002CB (         715.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX3               00007A00 00007CCA 000002CB (         715.) PAGE 9
TEST 004                                 00007E00 000089BF 00000BC0 (        3008.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX4               00007E00 000089BF 00000BC0 (        3008.) PAGE 9
TEST 005                                 00008A00 00009096 00000697 (        1687.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX4               00008A00 00009096 00000697 (        1687.) PAGE 9
TEST 006                                 00009200 000095CE 000003CF (         975.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX5               00009200 000095CE 000003CF (         975.) PAGE 9
TEST_007                                 00009600 0000979D 0000019E (         414.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX5               00009600 0000979D 0000019E (         414.) PAGE 9
TEST_008                                 00009800 00009B12 00000313 (         787.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX6               00009800 00009B12 00000313 (         787.) PAGE 9
TEST 009                                 00009C00 0000A049 0000044A (        1098.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX7               00009C00 0000A049 0000044A (        1098.) PAGE 9
TEST_010                                 0000A200 0000C341 00002142 (        8514.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX7               0000A200 0000C341 00002142 (        8514.) PAGE 9
TEST_011                                 0000C400 0000C855 00000456 (        1110.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX7               0000C400 0000C855 00000456 (        1110.) PAGE 9
TEST_012                                 0000CA00 0000EB51 00002152 (        8530.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX7               0000CA00 0000EB51 00002152 (        8530.) PAGE 9
TEST_013                                 0000EC00 0000F02C 0000042D (        1069.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX8               0000EC00 0000F02C 0000042D (        1069.) PAGE 9
TEST 014                                 0000F200 0000F52F 00000330 (         816.) PAGE 9 NOPIC,USR,CON,REL,LCL,NOSHR,  EXE,  RD,NOWRT,NOVEC
                    ECKAX8               0000F200 0000F52F 00000330 (         816.) PAGE 9

                                                      +-----------------+
                                              ! Symbols By Name !
                                              +-----------------+

Symbol                          Value        Symbol                          Value        Symbol                          Value
------                          -----        ------                          -----        ------                          -----
$ENV                            00000001     KA$W_LATENCY                    0000003E     T10_S4                          0000A991-R
$MO                             00000001     KA$W_MEM_SIZE                   00000044     T10_S5                          0000AB80-R
ALL                             00000009     KA$W_PROGRESS                   00000040     T10_S6                          0000ACE1-R
A_BUFFER_BEGIN                  00000C00-R   KA$W_RESERVED                   00000038     T10_S7                          0000AEC6-R
A_BUFFER_END                    00000C04-R   KA$W_WCS                        00000036     T10_S8                          0000B019-R
A_HALT_BUFFER                   00000C10-R   KA_PTABLE                       00000D42-R   T10_S9                          0000B391-R
A_MCHK_BUFFER                   00000C18-R   L_ACT_ISP                       00000C38-R   T11_S1                          0000C535-R
A_PSLMNE                        00000CCC-R   L_ACT_KSP                       00000C34-R   T11_S2                          0000C638-R
A_PWR_BUFFER                    00000C08-R   L_ACT_PSL                       00000C3C-R   T11_S3                          0000C73B-R
A_WCS_BUFFER                    00000C20-R   L_ACT_SP                        00000C30-R   T12_S1                          0000CB3D-R
DEFAULT                         00000000     L_EXP_AP                        00000C78-R   T12_S10                         0000DEE4-R
ERROR_CHECK                     00002A39-R   L_EXP_FP                        00000C7C-R   T12_S11                         0000E26B-R
GPR_TABLE                       00000C48-R   L_EXP_ISP                       00000C88-R   T12_S12                         0000E5C8-R
HALT                            00000004     L_EXP_KSP                       00000C84-R   T12_S13                         0000E802-R
HARDCNT                         00000800-R   L_EXP_PSL                       00000C8C-R   T12_S14                         0000E99A-R
IPR                             00000006     L_EXP_SP                        00000C80-R   T12_S2                          0000CE4D-R
KA$B_ACC_TYPE                   00000042     L_ISP                           00000C44-R   T12_S3                          0000D03C-R
KA$B_NODE_ID                    00000046     L_KSP                           00000C40-R   T12_S4                          0000D19D-R
KA$B_RESERVED                   00000043     L_REV_CHECK                     00000C90-R   T12_S5                          0000D38C-R
KA$B_SID_TYPE                   0000003D     MANUAL                          00000001     T12_S6                          0000D4ED-R
KA$K_LEN                        0000004B     MCHK                            00000005     T12_S7                          0000D6D2-R
KA$L_FLAGS                      00000032     MODELIST                        00000D0D-R   T12_S8                          0000D825-R
KA$L_MEM_SIZE                   00000047     PART_INIT                       00000963-R   T12_S9                          0000DB9D-R
KA$L_SID                        0000003A     PCS                             00000008     T13_S1                          0000ECDF-R
KA$M_CHAR                       00000100     PCS$A_PCS_BUFFER                00000C28-R   T13_S2                          0000ED82-R
KA$M_CRC                        00000010     POWER_FAIL                      00000003     T13_S3                          0000EE20-R
KA$M_D_FLOAT                    00000002     REG_TABLE                       00000C94-R   T13_S4                          0000EEBE-R
KA$M_EDIT                       00000080     SPACE1                          000008CF-R   T14_S1                          0000F2EB-R
KA$M_F_FLOAT                    00000001     SPACE10                         00000905-R   T14_S2                          0000F38E-R
KA$M_G_FLOAT                    00000004     SPACE2                          000008D1-R   T14_S3                          0000F43A-R
KA$M_H_FLOAT                    00000008     SPACE3                          000008D4-R   T1_S1                           00007226-R
KA$M_PACKED                     00000020     SPACE4                          000008D8-R   T1_S2                           000072F6-R
KA$M_PACK_MUL                   00000040     SPACE5                          000008DD-R   T1_S3                           000073AE-R
KA$M_SB_ERR                     02000000     SPACE6                          000008E3-R   T1_S4                           00007485-R
KA$M_TODR                       00010000     SPACE7                          000008EA-R   T2_S1                           00007666-R
KA$M_WCS_LD                     01000000     SPACE8                          000008F2-R   T2_S2                           000076E4-R
KA$V_CHAR                       00000008     SPACE9                          000008FB-R   T2_S3                           00007770-R
KA$V_CRC                        00000004     SYS$ASCTIM                      00010248     T2_S4                           0000780F-R
KA$V_D_FLOAT                    00000001     SYS$GETTIM                      00010378     T3_S1                           00007A6D-R
KA$V_EDIT                       00000007     SYS$SETPRT                      00010430     T3_S2                           00007AEB-R
KA$V_F_FLOAT                    00000000     T10_S1                          0000A331-R   T3_S3                           00007B77-R
KA$V_G_FLOAT                    00000002     T10_S10                         0000B6D8-R   T3_S4                           00007C16-R
KA$V_H_FLOAT                    00000003     T10_S11                         0000BA5F-R   T4_S1                           00007E88-R
KA$V_PACKED                     00000005     T10_S12                         0000BDBC-R   T4_S2                           00007FAB-R
KA$V_PACK_MUL                   00000006     T10_S13                         0000BFF6-R   T4_S3                           00008219-R
KA$V_SB_ERR                     00000019     T10_S14                         0000C191-R   T4_S4                           000083A5-R
KA$V_TODR                       00000010     T10_S2                          0000A641-R   T4_S5                           00008528-R
KA$V_WCS_LD                     00000018     T10_S3                          0000A830-R   T4_S6                           000086A8-R
```

```
Symbol                              Value         Symbol                              Value         Symbol                              Value
------                              -----         ------                              -----         ------                              -----
T4_S7                               00008828-R
T5_S1                               00008A80-R
T5_S2                               00008BB2-R
T5_S3                               00008CE4-R
T5_S4                               00008E16-R
T5_S5                               00008F6C-R
T6_S1                               00009294-R
T6_S2                               0000932B-R
T6_S3                               000093BD-R
T6_S4                               00009453-R
T7_S1                               0000967A-R
T7_S2                               000096F9-R
T8_S1                               00009847-R
T8_S2                               00009926-R
T8_S3                               00009A1B-R
T9_S1                               00009D29-R
T9_S2                               00009E2C-R
T9_S3                               00009F2F-R
TEST_001                            00007224-R
TEST_001_X                          00007545-R
TEST_002                            00007624-R
TEST_002_X                          000078A5-R
TEST_003                            00007A20-R
TEST_003_X                          00007CB9-R
TEST_004                            00007E30-R
TEST_004_X                          000089B8-R
TEST_005                            00008A28-R
TEST_005_X                          0000908F-R
TEST_006                            00009218-R
TEST_006_X                          000095C7-R
TEST_007                            0000961C-R
TEST_007_X                          00009796-R
TEST_008                            00009830-R
TEST_008_X                          00009B0B-R
TEST_009                            00009C18-R
TEST_009_X                          0000A042-R
TEST_010                            0000A220-R
TEST_010_X                          0000C33A-R
TEST_011                            0000C424-R
TEST_011_X                          0000C84E-R
TEST_012                            0000CA2C-R
TEST_012_X                          0000EB4A-R
TEST_013                            0000EC18-R
TEST_013_X                          0000F025-R
TEST_014                            0000F21C-R
TEST_014 X                          0000F528-R
TU58                                00000002
WCS                                 00000007
```

```
Symbol                          Value       Symbol                          Value       Symbol                          Value
---- -                          -----       ------                          -----       ------                          -----



          Key for special characters above:
                +------------ ------+
                ! *  - Undefined    !
                ! U  - Universal    !
                ! R  - Relocatable  !
                ! X  - External     !
                ! V  - Vectored     !
                +-------------------+
                                                 +----------------+
                                                 ! Image Synopsis !
                                                 +----------------+

Virtual memory allocated:                         00000200 0000F5FF 0000F400 (62464. bytes, 122. pages)
Stack size:                                              0. pages
Image binary virtual block limits:                       1.      122. (  122. blocks)
Image name and identification:                    ECKAX 5.2
Number of files:                                        10.
Number of modules:                                       9.
Number of program sections:                             34.
Number of global symbols:                              210.
Number of image sections:                                1.
Image type:                                       SYSTEM.
Map format:                                       DEFAULT in file DISK$DIAGPACK02:[TAYLOR.ECKAX]ECKAX.MAP;3
Estimated map length:                             56. blocks
                                              +---------------------+
                                              ! Link Run Statistics !
                                              +---------------------+

Performance Indicators                            Page Faults   CPU Time        Elapsed Time
----------------------                            -----------   --------        ------------
    Command processing:                                   108   00:00:00.67     00:00:01.84
    Pass 1:                                               303   00:00:01.25     00:00:04.64
    Allocation/Relocation:                                 22   00:00:00.14     00:00:00.65
    Pass 2:                                               174   00:00:02.77     00:00:07.22
    Map data after object module synopsis:                  6   00:00:00.53     00:00:00.62
    Symbol table output:                                    0   00:00:00.02     00:00:0u.10
Total run values:                                         613   00:00:05.38     00:00:15.07

Using a working set limited to 1862 pages and 59 pages of data storage (excluding image)

Total number object records read (both passes):   722
    of which 14 were in libraries and 16 were DEBUG data records containing 1215 bytes

Number of modules extracted explicitly             = 0
    with 1 extracted to resolve undefined symbols

0 library searches were for symbols not in the library searched

A total of 0 global symbol table records was written

LINK/CONTIG/SYST=%X200/EXE=ECKAX/MAP=ECKAX ECKAX1+ECKAX2+ECKAX3+ECKAX4+ECKAX5+ECKAX6+ECKAX7+ECKAX8
```

Table of contents

```
    (1)        1        ECKAX1: HEADER MODULE
    (2)       62        DECLARATIONS
    (3)      171        Program Header Data Block
    (3)      185        Dispatch Table
    (3)      198        Hardware Parameter Table
    (4)      212        Program Global Data Section
    (5)      259        Program Text Section
    (6)      324        Initialization Procedure
    (8)      375        Clean-up Procedure
    (9)      442        Summary Report Procedure
    (11)     515        Summary Information Block Initialization Subroutine
    (13)     568        Summary Information Input Subroutine
```

```
        0000     1         .SBTTL  ECKAX1: HEADER MODULE
        0000     2         .TITLE  ECKAX1  VAX 11/750 Specific CPU Cluster Exerciser
        0000     3         .IDENT  /5.2/
        0000     4
        0000     5 ;
        0000     6 ; Copyright (C) 1980,1983,1986
        0000     7 ; Digital Equipment Corporation, Maynard, Massachusetts 01754
        0000     8 ;
        0000     9 ; This software is furnished  under a license for use only on a single
        0000    10 ; computer  system and  may be  copied only with  the inclusion of the
        0000    11 ; above copyright notice.  This software, or any other copies thereof,
        0000    12 ; may not be provided or  otherwise made available to any other person
        0000    13 ; except for use on such system and to one who agrees to these license
        0000    14 ; terms.  Title to and  ownership of the  software  shall at all times
        0000    15 ; remain in DEC.
        0000    16 ;
        0000    17 ; The information in this software is subject to change without notice
        0000    18 ; and should  not be construed  as a commitment  by Digital  Equipment
        0000    19 ; Corporation.
        0000    20 ;
        0000    21 ; DEC assumes  no  responsibility  for  the use or  reliability of its
        0000    22 ; software on equipment which is not supplied by DEC.
        0000    23 ;
        0000    24
        0000    25 ;++
        0000    26 ; Facility:      VAX Architecture Diagnostic.
        0000    27 ;
        0000    28 ; Abstract:      This module provides all the standard header informaton
        0000    29 ;                and routines required to interface to the VAX Diagnostic
        0000    30 ;                Supervisor.
        0000    31 ;
        0000    32 ; Environment:  VAX Diagnostic Supervisor.
        0000    33 ;
        0000    34 ; Author:        Fred Roemer      19-FEB-81       Version 1.4
        0000    35 ;
        0000    36 ;          NO CHANGES TO ANY MODULE VERSION 1.5 (DOCUMENT CHANGE ONLY)
        0000    37 ;
        0000    38 ;
        0000    39 ;          Dan Milleville  19-Mar-82                 Version 3.4
        0000    40 ;          Changed Test 3 exit code to accomodate temporarily a hardware problem.
        0000    41 ;
        0000    42 ;          Dave Shull      30-Nov-82                 Version 4.0
        0000    43 ;          Added tests 13 and 14 for coverage of PCS module.
        0000    44 ;
        0000    45 ;          Tai-Chun Pan    04-Mar-83                 Version 5.0
        0000    46 ;          Added test 5 for coverage of Processor Halts Architecture Undefined.
        0000    47 ;          Added tests 11 and 12 for coverage of new ROM with MRSP of TU58.
        0000    48 ;
        0000    49 ;          Dan Milleville  21-Nov-83                 Version 5.1
        0000    50 ;          Fixed error in timer routine (didn't turn off timer).
        0000    51 ;
        0000    52 ;          Wayne R. Taylor February 1986             Version 5.2
        0000    53 ;          Redefine the IPR's to jive with starlet.
        0000    54 ;          Modified the code for problem report Eckax5 where:
        0000    55 ;                  Tests 6 & 13 tests with cache off.
```

```
0000    56 ;            Subtest 1 does not check bit 0 of the memory under test.
0000    57 ;            Subtest 4 tests row 0, 1, 2, but not row 3.
0000    58 ;        Modifications were supplied by the submitter.
0000    59 ;        Search on 5.2.
0000    60 ;--
```

```
                         0000     62          .SBTTL  DECLARATIONS
                         0000     63          .LIST   MEB
                         0000     64          .NLIST  CND
                     00000000     65          .PSECT  ECKAX1, PAGE, NOWRT
                         0000     66          .LIBRARY        SYS$LIBRARY:DIAG ; VAX family diagnostic library
                         0000     67          $DS_BGNMOD      CEP_REPAIR
                         0000        ;;;      Macro-32 Diagnostic macro library Version 8.3  DIAG.MLB (245)
                         0000     68
                         0000     69 ;
                         0000     70 ; Include files:
                         0000     71 ;
                         0000     72          .LIBRARY         ECKAX           ; CPU Cluster Exerciser library.
                         0000     73
                         0000     77          $PR750DEF        ; 5.2
                         0000     78          $DS_SCBDEF
                         0000     79          $DS_DSDEF
                         0000     80          $DS_DSSDEF
                         0000     81          $DS_DSADEF
                         0000     82          $DS_PARDEF
                         0000     83          $DS_PSLDEF
                         0000     84          $PRDEF
                         0000     85          $DS_HPODEF
                         0000     86          KA_DEF
                         0000     87
                         0000     88 ;
                         0000     89 ; Own storage and equated symbols:
                         0000     90 ;
            00000004     0000     92 A_BUFFER_BEGIN::         .BLKL    1        ; address of data buffer
            00000008     0004     93 A_BUFFER_END::           .BLKL    1
            00000010     0008     94 A_PWR_BUFFER::           .BLKL    2        ; address of POWER FAIL data buffer
            00000018     0010     95 A_HALT_BUFFER::          .BLKL    2        ; address of HALTS data buffer
            00000020     0018     96 A_MCHK_BUFFER::          .BLKL    2        ; address of MACHINE CHECK data buffer
            00000028     0020     97 A_WCS_BUFFER::           .BLKL    2        ; address of WCS data buffer
            00000030     0028     98 PCS$A_PCS_BUFFER::       .BLKL    2        ; address of PCS data buffer
                         0030     99
            00000034     0030    100 L_ACT_SP::               .BLKL             ; used to store actual SP
            00000038     0034    101 L_ACT_KSP::              .BLKL             ; used to store actual KSP
            0000003C     0038    102 L_ACT_ISP::              .BLKL             ; used to store actual ISP
            00000040     003C    103 L_ACT_PSL::              .BLKL             ; used to store actual PSL
            00000044     0040    104 L_KSP::                  .BLKL             ; used to store initial KSP
            00000048     0044    105 L_ISP::                  .BLKL             ; used to store initial ISP
                         0048    106
                         0048    107 GPR_TABLE::
            00000000     0048    108         .LONG    ^X00000000          ; data for register 0
            11111111     004C    109         .LONG    ^X11111111          ; data for register 1
            22222222     0050    110         .LONG    ^X22222222          ; data for register 2
            33333333     0054    111         .LONG    ^X33333333          ; data for register 3
            44444444     0058    112         .LONG    ^X44444444          ; data for register 4
            55555555     005C    113         .LONG    ^X55555555          ; data for register 5
            66666666     0060    114         .LONG    ^X66666666          ; data for register 6
            77777777     0064    115         .LONG    ^X77777777          ; data for register 7
            88888888     0068    116         .LONG    ^X88888888          ; data for register 8
            99999999     006C    117         .LONG    ^X99999999          ; data for register 9
            AAAAAAAA     0070    118         .LONG    ^XAAAAAAAA          ; data for register A
```

```
                                BBBBBBBB  0074    119             .LONG    ^XBBBBBBBB             ; data for register B
                                0000007C  0078    120 L_EXP_AP::       .BLKL             ; location to store argument counter
                                00000080  007C    121 L_EXP_FP::       .BLKL             ; location to store frame pointer
                                00000084  0080    122 L_EXP_SP::       .BLKL             ; location to store expected SP
                                00000088  0084    123 L_EXP_KSP::      .BLKL             ; location to store expected KSP
                                0000008C  0088    124 L_EXP_ISP::      .BLKL             ; location to store expected ISP
                                00000090  008C    125 L_EXP_PSL::      .BLKL             ; location to store expected PSL
                                          0090    126
                                0000005E  0090    127 L_REV_CHECK::    .LONG    ^X5E     ; This represents the revision of microcode
                                          0094    128                                    ; before the PCS/WCS module (L0008) replaced
                                          0094    129                                    ; the L0005 module
                                          0094    130
                                          0094    131
                                          0094    132 REG_TABLE::
                             20 30 52 00' 0094    133             .ASCIC   R0
                                       03 0094
                             20 31 52 00' 0098    134             .ASCIC   R1
                                       03 0098
                             20 32 52 00' 009C    135             .ASCIC   R2
                                       03 009C
                             20 33 52 00' 00A0    136             .ASCIC   R3
                                       03 00A0
                             20 34 52 00' 00A4    137             .ASCIC   R4
                                       03 00A4
                             20 35 52 00' 00A8    138             .ASCIC   R5
                                       03 00A8
                             20 36 52 00' 00AC    139             .ASCIC   R6
                                       03 00AC
                             20 37 52 00' 00B0    140             .ASCIC   R7
                                       03 00B0
                             20 38 52 00' 00B4    141             .ASCIC   R8
                                       03 00B4
                             20 39 52 00' 00B8    142             .ASCIC   R9
                                       03 00B8
                             30 31 52 00' 00BC    143             .ASCIC   R10
                                       03 00BC
                             31 31 52 00' 00C0    144             .ASCIC   R11
                                       03 00C0
                             20 50 41 00' 00C4    145             .ASCIC   AP
                                       03 00C4
                             20 50 46 00' 00C8    146             .ASCIC   FP
                                       03 00C8
                                          00CC    147
                                          00CC    148
                                          00CC    149 ; The following is used to intrepret the PSL and to print out an English
                                          00CC    150 ; language description of its contents.
                                          00CC    151
                                          00CC    152 A_PSLMNE::
44 50 46 2C 2C 2C 50 54 2C 4D 43 00'      00CC    153             .ASCIC   ~CM,TP,,,FPD,IS,CUR=2a,PRV=2a,, ~
50 2C 40 32 3D 52 55 43 2C 53 49 2C       00D8
35 3D 4C 50 49 2C 2C 40 32 3D 56 52       00E4
44 2C 2C 2C 2C 2C 2C 2C 2C 2C 58 5E       00F0
2C 4E 2C 54 2C 56 49 2C 55 46 2C 56       00FC
                      43 2C 56 2C 5A      0108
                                   40     00CC
```

```
                                     010D   154                    IPL=5^X,........,DV,FU,IV,T,N,Z,V,C
                                     010D   155
                           00000004  010D   156 MODELIST::      .LONG   4
                           00000121' 0111   157          .ADDRESS        AKM
                           00000128' 0115   158          .ADDRESS        AEM
                           00000132' 0119   159          .ADDRESS        ASM
                           0000013D' 011D   160          .ADDRESS        AUM
                                     0121   161
              4C 45 4E 52 45 4B 00'  0121   162 AKM:     .ASCIC   KERNEL
                                06   0121
     45 56 49 54 55 43 45 58 45 00'  0128   163 AEM:     .ASCIC   EXECUTIVE
                                09   0128
  52 4F 53 49 56 52 45 50 55 53 00'  0132   164 ASM:     .ASCIC   SUPERVISOR
                                0A   0132
                    52 45 53 55 00'  013D   165 AUM:     .ASCIC   USER
                                04   013D
                                     0142   166
                                     0142   171         .SBTTL  Program Header Data Block
                                     0142   172 ;++
                                     0142   173 ; Functional Description:
                                     0142   174 ;
                                     0142   175 ;       The program header data block contains the parameters which
                                     0142   176 ;       allow the Diagnostic Supervisor to control the program.
                                     0142   177 ;       The Diagnostic Supervisor looks for the header information
                                     0142   178 ;       beginning at virtual address 200(hex).
                                     0142   179 ;
                                     0142   180 ;--
                                     0142   181         $DS_HEADER  <VAX 11/750 Specific CPU Cluster Exerciser - ZZ-ECKAX-5.2 >, -
                                     0142   182                         REV = 5, UPDATE = 2, -
                                     0142   183                         NUNIT = 1
                                     0142                       .SAVE
                                 00000000                       .PSECT  $HEADER, PAGE, NOEXE, NOWRT
                           00000058' 0000       L$L_HEADLENGTH: .LONG    A_HEADEND-.     ; LENGTH OF THE HEADER DATA BLOCK
                           00000005  0004       L$L_ENVIRON:    .LONG    <ENV$_SUPER@ENV$V_SUPER>+$ENV  ; PROGRAM ENVIRONMENT
                           00000058' 0008       L$A_NAME:       .ADDRESS T_NAME               ; PROGRAM NAME TEXT ADDRESS
                           00000005  000C       L$L_REV:        .LONG    5                    ; PROGRAM REVISION LEVEL
                           00000002  0010       L$L_UPDATE:     .LONG    2                    ; DIAGNOSTIC ENGINEERING PATCH ORDER
                           00000000' 0014       L$A_LASTAD:     .ADDRESS LASTAD               ; FIRST FREE LOCATION AFTER PROGRAM
                           00000000' 0018       L$A_DTP:        .ADDRESS DISPATCH             ; TEST DISPATCH TABLE POINTER
                           00000084' 001C       L$A_DEVP:       .ADDRESS AL_DEVTYP            ; DEVICE TYPE LIST POINTER
                           00000001  0020       L$L_UNIT:       .LONG    1                    ; NUMBER OF UNITS THAT CAN BE TESTED
                           00000000' 0024       L$A_DREG:       .ADDRESS DEV_REG              ; DEVICE REGISTER CONTENTS TABLE POINTER
                           0000001C  0028                       .LONG   28                    ; Ptable length for 4.05 compatability
00000000'00000000'00000000'00000000' 002C                       .LONG   0[4]                  ; UNUSED OFFSETS
                           00000000' 003C       L$A_ICP:        .ADDRESS INITIALIZE           ; INITIALIZATION CODE POINTER
                           00000000' 0040       L$A_CCP:        .ADDRESS CLEAN_UP             ; CLEAN-UP CODE POINTER
                           00000000' 0044       L$A_REPP:       .ADDRESS SUMMARY              ; SUMMARY REPORT CODE POINTER
                           00000000' 0048       L$A_STATAB:     .ADDRESS 0                    ; STATISTIC TABLE POINTER
                           00000000  004C       L$L_ERRTYP:     .LONG    0                    ; # OF TYPES OF $ERRSOFT & $ERRPREP
                           00000051' 0050       L$A_SECNAM:     .ADDRESS SECTION              ; LIST OF SECTION NAME ADDRESSES
                           00000000' 0054       L$A_TSTCNT:     .ADDRESS L_TSTCNT             ; POINTER TO NUMBER OF TESTS
                                     0058       A_HEADEND:
20 30 35 37 2F 31 31 20 58 41 56 00' 0058       T_NAME:         .ASCIC  \VAX 11/750 Specific CPU Cluster Exerciser - ZZ-ECKAX-5.2 \
55 50 43 20 63 69 66 69 63 65 70 53  0064
65 78 45 20 72 65 74 73 75 6C 43 20  0070
2D 5A 5A 20 2D 20 72 65 73 69 63 72  007C
      20 32 2E 35 2D 58 41 4B 43 45  0088
                                 39  0058
                                 00000000                       .PSECT  _LAST, PAGE
                                     0000       LASTAD:
                                 00000000                       .PSECT  $TSTCNT, NOEXE, NOWRT, OVR, LONG
                                     0000       L_TSTCNT:
                                 00000142                       .RESTORE
                                 0142   185           .SBTTL  Dispatch Table
                                 0142   186 ;++
                                 0142   187 ; Functional Description:
                                 0142   188 ;
                                 0142   189 ;           The Dispatch Table is a collection of addresses generated at the
                                 0142   190 ;           beginning of each test and grouped together into a contiguous
                                 0142   191 ;           list by the linker.  This is done by defining a PSECT called
                                 0142   192 ;           DSIPATCH.
                                 0142   193 ;
                                 0142   194 ;--
                                 0142   195
                                 0142   196           $DS_DISPATCH
                                 0142                         .SAVE
                             00000000                         .PSECT  DISPATCH, LONG, NOWRT, NOEXE
                                 0000        DISPATCH:
                             00000000                         .PSECT  DISPATCH_X, LONG, NOWRT, NOEXE
00000000'00000000'00000000'00000000' 0000                     .LONG   0[6]
                  00000000'00000000' 0010
                             00000142                         .RESTORE
                                 0142   197
                                 0142   198           .SBTTL  Hardware Parameter Table
                                 0142   199 ;++
                                 0142   200 ;
                                 0142   201 ;           The hardware parameter table holds all of the hardware related
                                 0142   202 ;           parameters which can be modified by the operator at run time.
                                 0142   203 ;           The Daignostic Supervisor maintains a seperate copy of the
                                 0142   204 ;           hardware parameter table for each unit under test.
                                 0142   205 ;           The hardware parameter code section is used to actually supply
                                 0142   206 ;           the values to these locations.
                                 0142   207 ;--
                                 0142   208
                     00000146    0142   209 KA_PTABLE::     .BLKL   1               ; pointer to the CPU parameter table.
                                 0146   210
```

```
                      0146   212         .SBTTL  Program Global Data Section
                      0146   213 ;++
                      0146   214 ;
                      0146   215 ; General data used by more than one module.
                      0146   216 ;
                      0146   217 ;--
                      0146   218         .SAVE
                  00000000   219         .PSECT  DATA,PAGE
                      0000   220
                      0000   221 DEV_REG:
                      0000   222
                      0000   223 ;
                      0000   224 ;       Miscellaneous data
                      0000   225 ;
                      0000   226
            00000000  0000   227 HARDCNT::       .LONG   0                 ; Exerciser summary hard
                      0004   228                                           ;  error count.
            0000000C  0004   229 Q_TEST_TIME:    .BLKQ   1                 ; Test start time, used to calculate
                      000C   230                                           ;  test run time for summary report.
                      000C   231
                      000C   232
                      000C   233 ;+
                      000C   234 ; Test summary information data block
                      000C   235 ;
                      000C   236 ;       The test summary information block is dynamically allocated, based
                      000C   237 ;       on the number of tests in the program.  There is one entry per test
                      000C   238 ;       which contains all the pertinent error and run time information.
                      000C   239 ;       This information is displayed by the SUMMARY procedure, which is
                      000C   240 ;       invoked at the end of a program run or by an operator 'SUMMARY'
                      000C   241 ;       command.
                      000C   242 ;
                      000C   243 ;--
                      000C   244
            00000014  000C   245 SUM_A_BLOCK:    .BLKL   2                 ; Two longword array to receive the
                      0014   246                                           ;  start and end addresses of the
                      0014   247                                           ;  summary information data block.
                      0014   248         .SAVE
                      0014   249         .PSECT  $ABS$
            00000000  0610   250         . = 0
                      0000   251 $DEF    SUM_L_HARDCNT, .BLKL, 1           ; Hard error count entry displacement.
                      0000           .IIF    NB,SUM_L_HARDCNT,         SUM_L_HARDCNT:
            00000004  0000           .IIF    NB,.BLKL,         .BLKL   1
                      0004   252 $DEF    SUM_Q_RUNTIME, .BLKQ, 1           ; Run time entry displacement.
                      0004           .IIF    NB,SUM_Q_RUNTIME,         SUM_Q_RUNTIME:
            0000000C  0004           .IIF    NB,.BLKQ,         .BLKQ   1
                      000C   253 $DEF    SUM_K_SIZE                        ; Size of summary information block
                      000C           .IIF    NB,SUM_K_SIZE,  SUM_K_SIZE:
                      000C   254                                           ;  for one test.
                  00000014   255         .RESTORE
                      0014   256
            00000018  0014   257 SUM_BLK_SIZE:   .BLKL   1                 ; Number of pages for summary info block.
```

```
                                     0018    259         .SBTTL  Program Text Section
                                     0018    260  ;++
                                     0018    261  ;
                                     0018    262  ;          This section contains all the common character string type data
                                     0018    263  ;          entries.
                                     0018    264  ;
                                     0018    265  ;--
                                     0018    266
                                     0018    267  ;+
                                     0018    268  ; Program section names:
                                     0018    269  ;                              MANUAL,TU58,POWER_FAIL,HALT,MCHK,IPR,WCS,PCS,ALL
                                     0018    270  ;
                                     0018    271  ;-
                                     0018    272
                                     0018    273         $DS_SECTION     MANUAL,TU58,POWER_FAIL,HALT,MCHK,IPR,WCS,PCS,ALL
                                     0018         S_DEFAULT:
               54 4C 55 41 46 45 44 00' 0018                          .ASCIC  \DEFAULT\
                                 07  0018
                                     0020         S_MANUAL:
                  4C 41 55 4E 41 4D 00' 0020                          .ASCIC  \MANUAL\
                                 06  0020
                                     0027         S_TU58:
                        38 35 55 54 00' 0027                          .ASCIC  \TU58\
                                 04  0027
                                     002C         S_POWER_FAIL:
      4C 49 41 46 5F 52 45 57 4F 50 00' 002C                          .ASCIC  \POWER_FAIL\
                                 0A  002C
                                     0037         S_HALT:
                        54 4C 41 48 00' 0037                          .ASCIC  \HALT\
                                 04  0037
                                     003C         S_MCHK:
                        4B 48 43 4D 00' 003C                          .ASCIC  \MCHK\
                                 04  003C
                                     0041         S_IPR:
                           52 50 49 00' 0041                          .ASCIC  \IPR\
                                 03  0041
                                     0045         S_WCS:
                           53 43 57 00' 0045                          .ASCIC  \WCS\
                                 03  0045
                                     0049         S_PCS:
                           53 43 50 00' 0049                          .ASCIC  \PCS\
                                 03  0049
                                     004D         S_ALL:
                           4C 4C 41 00' 004D                          .ASCIC  \ALL\
                                 03  004D
                                     0051         SECTION:
                           0000000A  0051                          .LONG   10
                           00000018' 0055                          .LONG   S_DEFAULT
                           00000020' 0059                          .LONG   S_MANUAL
                           00000027' 005D                          .LONG   S_TU58
                           0000002C' 0061                          .LONG   S_POWER_FAIL
                           00000037' 0065                          .LONG   S_HALT
                           0000003C' 0069                          .LONG   S_MCHK
                           00000041' 006D                          .LONG   S_IPR
                           00000045' 0071                          .LONG   S_WCS
```

```
                      00000049' 0075                      .LONG   S_PCS
                      0000004D' 0079                      .LONG   S_ALL
                                007D   274
                                007D   275
                                007D   276
                                007D   277
                                007D   278
                                007D   279 ;+
                                007D   280 ; Device mnemonics list:
                                007D   281 ;       KA750 = VAX-11/750 CPU
                                007D   282 ;-
                                007D   283         $DS_DEVTYP      KA750
               30 35 37 41 4B 00' 007D     T_KA750:                .ASCIC  \KA750\
                             05 007D
                             00 0083             .BYTE   0
                       00000001 0084     AL_DEVTYP:      .LONG   $$N
                      0000007D' 0088                     .LONG   T_KA750
                                008C   284
                                008C   285
                                008C   286 ;+
                                008C   287 ; Names of device registers:  None.
                                008C   288 ;-
                                008C   289
                                008C   290
                                008C   291 ;+
                                008C   292 ; ASCII buffers
                                008C   293 ;-
                       0000000B 008C   294 BUF_SIZE = 11
                                008C   295 QWD_TIME:
                      0000 000B 008C   296         .WORD   BUF_SIZE, 0
                      00000094' 0090   297         .LONG   T_TIME
                                0094   298
                       0000009F 0094   299 T_TIME: .BLKB   BUF_SIZE
                                009F   300
                                009F   301
                                009F   302 ;+
                                009F   303 ; Formatted ASCII output statements
                                009F   304 ;-
                                009F   305
                                009F   306 FMT_SUMMARY:                    ; FAO message for module summary printout.
20 57 5A 21 20 74 73 65 54 2F 21 00' 009F   307         .ASCIC   !/Test !ZW had !ZL hard error!%s, !AS run time.
64 72 61 68 20 4C 5A 21 20 64 61 68  00AB
21 20 2C 73 25 21 72 6F 72 72 65 20  00B7
2E 65 6D 69 74 20 6E 75 72 20 53 41  00C3
                                 2F  009F
                                00CF   308
                                00CF   309 ;+
                                00CF   310 ; Global ASCII strings
                                00CF   311 ;-
                                00CF   312
                          20 00' 00CF   313 SPACE1::        .ASCIC  ' '
                             01  00CF
                       20 20 00' 00D1   314 SPACE2::        .ASCIC  '  '
                             02  00D1
                    20 20 20 00' 00D4   315 SPACE3::        .ASCIC  '   '
```

```
                             03  00D4
                 20 20 20 20 00' 00D8   316 SPACE4::        .ASCIC  '    '
                             04  00D8
              20 20 20 20 20 00' 00DD   317 SPACE5::        .ASCIC  '     '
                             05  00DD
           20 20 20 20 20 20 00' 00E3   318 SPACE6::        .ASCIC  '      '
                             06  00E3
        20 20 20 20 20 20 20 00' 00EA   319 SPACE7::        .ASCIC  '       '
                             07  00EA
     20 20 20 20 20 20 20 20 00' 00F2   320 SPACE8::        .ASCIC  '        '
                             08  00F2
  20 20 20 20 20 20 20 20 20 00' 00FB   321 SPACE9::        .ASCIC  '         '
                             09  00FB
20 20 20 20 20 20 20 20 20 20 00' 0105  322 SPACE10::       .ASCIC  '          '
                             0A  0105
```

```
0110                 .SBTTL  Initialization Procedure
0110          ;++
0110          ; Functional description:
0110          ;
0110          ;       This routine will be executed at the beginning of each pass.
0110          ;       This section is responsible for control of:
0110          ;
0110          ;       A) a double iteration of the exerciser to make up 1 pass;
0110          ;       B) test summary information block initialization.
0110          ;
0110          ; Calling sequence:
0110          ;
0110          ;       The Diagnostic Supervisor automatically invokes this routine with
0110          ;       a procedure call instruction.
0110          ;
0110          ; Input parameters:
0110          ;
0110          ;       INTERS_FLAG = 0 representing 1st iteration of a double iteration
0110          ;               of the dispatch table to make up 1 pass.
0110          ;
0110          ; Implicit inputs:      None
0110          ;
0110          ; Output parameters:    None
0110          ;
0110          ; Implicit outputs:     None
0110          ;
0110          ; Completion codes:     None
0110          ;
0110          ; Side effects:         None
0110          ;
0110          ;--
```

```
                                     0110   356         $DS_BGNINIT
                                     0110                       .SAVE
                                 00000000                       .PSECT  INITIALIZE,LONG
                                     0000       INITIALIZE:
                               0000  0000                       .WORD   ^M<>            ; ENTRY MASK
                   00000000'EF   D4  0002   357         CLRL    HARDCNT                         ; Initialize the test summary
                                     0008   358                                                 ;  hard errors counter.
                                     0008   359         $DS_BPASS0      5$
                            1D   E0  0008                       BBS     #DSA$V_PASS0,-  ; BR IF PASS 0
            03 0000FE00 9F           000A                               @#DSA$GL_FLAGS, 5$
                          0087   31  0010   360         BRW             10$
                                     0013   361 5$:
                                     0013   362         $DS_GPHARD_S    #0, KA_PTABLE           ; Get CPU parameter table.
                   00000142'EF   DF  0013                       PUSHAL  KA_PTABLE
                            00   DD  0019                       PUSHL   #0
               00010018 9F  02   FB  001B                       CALLS   #2, @#DS$GPHARD
                   00000110'EF   16  0022   363         JSB     SUMMARY_INIT                    ; Structure module summary
                                     0028   364                                                 ; Information block
                                     0028   365         $DS_GETBUF_S    PAGCNT=#3, RETADR=A_BUFFER_BEGIN
                            00   DD  0028                       PUSHL   #0
                            00   DD  002A                       PUSHL   #0
                   00000000'EF   7F  002C                       PUSHAQ  A_BUFFER_BEGIN
                            03   DD  0032                       PUSHL   #3
               00010120 9F  04   FB  0034                       CALLS   #4, @#DS$GETBUF
                                     003B   366         $DS_GETBUF_S    PAGCNT=#1, RETADR=A_PWR_BUFFER
                            00   DD  003B                       PUSHL   #0
                            00   DD  003D                       PUSHL   #0
                   00000008'EF   7F  003F                       PUSHAQ  A_PWR_BUFFER
                            01   DD  0045                       PUSHL   #1
               00010120 9F  04   FB  0047                       CALLS   #4, @#DS$GETBUF
                                     004E   367         $DS_GETBUF_S    PAGCNT=#1, RETADR=A_HALT_BUFFER
                            00   DD  004E                       PUSHL   #0
                            00   DD  0050                       PUSHL   #0
                   00000010'EF   7F  0052                       PUSHAQ  A_HALT_BUFFER
                            01   DD  0058                       PUSHL   #1
               00010120 9F  04   FB  005A                       CALLS   #4, @#DS$GETBUF
                                     0061   368         $DS_GETBUF_S    PAGCNT=#1, RETADR=A_MCHK_BUFFER
                            00   DD  0061                       PUSHL   #0
                            00   DD  0063                       PUSHL   #0
                   00000018'EF   7F  0065                       PUSHAQ  A_MCHK_BUFFER
                            01   DD  006B                       PUSHL   #1
               00010120 9F  04   FB  006D                       CALLS   #4, @#DS$GETBUF
                                     0074   369         $DS_GETBUF_S    PAGCNT=#20, RETADR=A_WCS_BUFFER
                            00   DD  0074                       PUSHL   #0
                            00   DD  0076                       PUSHL   #0
                   00000020'EF   7F  0078                       PUSHAQ  A_WCS_BUFFER
                            14   DD  007E                       PUSHL   #20
               00010120 9F  04   FB  0080                       CALLS   #4, @#DS$GETBUF
                                     0087   370         $DS_GETBUF_S    PAGCNT=#10, RETADR=PCS$A PCS_BUFFER
                            00   DD  0087                       PUSHL   #0
                            00   DD  0089                       PUSHL   #0
                   00000028'EF   7F  008B                       PUSHAQ  PCS$A_PCS_BUFFER
                            0A   DD  0091                       PUSHL   #10
               00010120 9F  04   FB  0093                       CALLS   #4, @#DS$GETBUF
                                     009A   371 10$:
```

```
                                     009A   372        $DS_ENDPASS_G
              00010010 9F   6E   FA  009A                      CALLG   (SP), @#DS$ENDPASS
                                     00A1   373        $DS_ENDINIT
                                     00A1       INITIALIZE_X:
              00010058 9F   6E   FA  00A1                      CALLG   (SP), @#DS$BREAK
                                 04  00A8                      RET                     ; RETURN TO DIAGNOSTIC SUPERVISOR
                                 00000110                      .RESTORE
```

```
                              0110   375          .SBTTL  Clean-up Procedure
                              0110   376 ;++
                              0110   377 ; Functional description:
                              0110   378 ;
                              0110   379 ;        This routine is executed at the completion of a program run.
                              0110   380 ;        This section is responsible for:
                              0110   381 ;
                              0110   382 ;        A) reinitialization of flags controlling double iteration per pass;
                              0110   383 ;        B) activation of a statistical summary report ($DS_SUMMARY_G);
                              0110   384 ;        C) releasing all temporary buffers.
                              0110   385 ;        D) clearing of IPL
                              0110   386 ;
                              0110   387 ; Calling sequence:
                              0110   388 ;
                              0110   389 ;        The Diagnostic Supervisor automatically invokes this routine with
                              0110   390 ;        a procedure call instruction.
                              0110   391 ;
                              0110   392 ; Input parameters:     None
                              0110   393 ;
                              0110   394 ; Implicit inputs:
                              0110   395 ;
                              0110   396 ;        The summary information block to be invoked by the $DS_SUMMARY_G
                              0110   397 ;        call has been dynamically filled at run time.
                              0110   398 ;
                              0110   399 ; Output parameters:    None
                              0110   400 ;
                              0110   401 ; Implicit outputs:     None
                              0110   402 ;
                              0110   403 ; Completion codes:     None
                              0110   404 ;
                              0110   405 ; Side effects:
                              0110   406 ;
                              0110   407 ;        The test summary information is destroyed after it is reported.
                              0110   408 ;
                              0110   409 ;--
                              0110   410
                              0110   411          $DS_BGNCLEAN
                              0110                       .SAVE
                          00000000                       .PSECT  CLEANUP,LONG
                              0000       CLEAN_UP:
                        0000  0000                       .WORD   ^M<>              ; ENTRY MASK
              55    3E    DB  0002   412          MFPR    #PR$_SID,R5                    ; get the SID register
      56   55    08    08    EF  0005   413          EXTZV   #8,#8,R5,R6                    ; get the micro revision level
       00000090'8F    55    D1  000A   414          CMPL    R5,#L_REV_CHECK                ; check to see if this is a
                              0011   415                                                 ; PCS module
                    07    1B  0011   416          BLEQU   NOT_PCS                        ; br = not PCS module
                              0013   417          $DS_LOADPCS_S                          ; re-load the microcode and
       000101C0 9F    00    FB  0013                       CALLS   #0, @#DS$LOADPCS
                              001A   418                                                 ; patch bits
                              001A   419 NOT_PCS:
                              001A   420          $DS_CLRVEC_S    VECTOR=#SCB$L_TIMER
          000000C0 8F          DD  001A                       PUSHL   #SCB$L_TIMER
       00010168 9F    01    FB  0020                       CALLS   #1, @#DS$CLRVEC
                              0027   421          $DS_CLRVEC_S    VECTOR=#SCB$L_POWER
                    0C    DD  0027                       PUSHL   #SCB$L_POWER
```

```
00010168 9F   01   FB  0029                          CALLS   #1, @#DS$CLRVEC
                       0030   422      $DS_CLRVEC_S     VECTOR=#SCB$L_CHME
   00000044 8F   DD  0030                          PUSHL   #SCB$L_CHME
00010168 9F   01   FB  0036                          CALLS   #1, @#DS$CLRVEC
                       003D   423      $DS_CLRVEC_S     VECTOR=#SCB$L_CHMS
   00000048 8F   DD  003D                          PUSHL   #SCB$L_CHMS
00010168 9F   01   FB  0043                          CALLS   #1, @#DS$CLRVEC
                       004A   424      $DS_CLRVEC_S     VECTOR=#SCB$L_CHMU
   0000004C 8F   DD  004A                          PUSHL   #SCB$L_CHMU
00010168 9F   01   FB  0050                          CALLS   #1, @#DS$CLRVEC
                       0057   425      $DS_CLRVEC_S     VECTOR=#SCB$L_CHMK
   00000040 8F   DD  0057                          PUSHL   #SCB$L_CHMK
00010168 9F   01   FB  005D                          CALLS   #1, @#DS$CLRVEC
                       0064   426      $DS_CLRVEC_S     VECTOR=#SCB$L_MACHCHK
              04   DD  0064                          PUSHL   #SCB$L_MACHCHK
00010168 9F   01   FB  0066                          CALLS   #1, @#DS$CLRVEC
                       006D   427      $DS_CLRVEC_S     VECTOR=#SCB$L_OPCCUS
              14   DD  006D                          PUSHL   #SCB$L_OPCCUS
00010168 9F   01   FB  006F                          CALLS   #1, @#DS$CLRVEC
                       0076   428      $DS_CLRVEC_S     VECTOR=#SCB$L_OPCDEC
              10   DD  0076                          PUSHL   #SCB$L_OPCDEC
00010168 9F   01   FB  0078                          CALLS   #1, @#DS$CLRVEC
                       007F   429      $DS_CLRVEC_S     VECTOR=#SCB$L_TXDB
   000000FC 8F   DD  007F                          PUSHL   #SCB$L_TXDB
00010168 9F   01   FB  0085                          CALLS   #1, @#DS$CLRVEC
                       008C   430      $DS_CLRVEC_S     VECTOR=#SCB$L_RXDB
   000000F8 8F   DD  008C                          PUSHL   #SCB$L_RXDB
00010168 9F   01   FB  0092                          CALLS   #1, @#DS$CLRVEC
                       0099   431      $DS_MMOFF_G
00010158 9F   6E   FA  0099                          CALLG   (SP), @#DS$MMOFF
                       00A0   432      $DS_INITSCB_G
00010170 9F   6E   FA  00A0                          CALLG   (SP), @#DS$INITSCB
                       00A7   433      $DS_SUMMARY_G
00010028 9F   6E   FA  00A7                          CALLG   (SP), @#DS$SUMMARY
                       00AE   434      $DS_RELBUF_S     PAGCNT=#26        ; release all data buffers
              00   DD  00AE                          PUSHL   #0
              00   DD  00B0                          PUSHL   #0
              1A   DD  00B2                          PUSHL   #26
00010128 9F   03   FB  00B4                          CALLS   #3, @#DS$RELBUF
                       00BB   435      $DS_RELBUF_S     SUM_BLK_SIZE      ; Release summary data block.
              00   DD  00BB                          PUSHL   #0
              00   DD  00BD                          PUSHL   #0
   00000014'EF   DD  00BF                          PUSHL   SUM_BLK_SIZE
00010128 9F   03   FB  00C5                          CALLS   #3, @#DS$RELBUF
          1C  00   DA  00CC   436      MTPR    #0,#PR750$_CSRS          ; disable TU58 interrupt 5.2
          1E  00   DA  00CF   437      MTPR    #0,#PR750$_CSTS          ; disable TU58 interrupt 5.2
          25  00   DA  00D2   438      MTPR    #0,#PR750$_CADR          ; make sure cache is enabled on cleanup 5.2
                       00D5   439      $DS_SETIPL_S     LEVEL=#0          ; clear IPL
              00   DD  00D5                          PUSHL   #0
00010178 9F   01   FB  00D7                          CALLS   #1, @#DS$SETIPL
                       00DE   440      $DS_ENDCLEAN
                       00DE        CLEAN_UP_X:
00010058 9F   6E   FA  00DE                          CALLG   (SP), @#DS$BREAK
                   04  00E5                          RET                       ; RETURN TO DIAGNOSTIC SUPERVISOR
              00000110                               .RESTORE
```

```
     0110    442          .SBTTL  Summary Report Procedure
     0110    443 ;++
     0110    444 ; Functional description:
     0110    445 ;
     0110    446 ;        This routine issues a summary report upon completion of a run or
     0110    447 ;        upon request from the operator.
     0110    448 ;        Information is taken from the 'head' of the test summary
     0110    449 ;        queue (SUMMARY_INFO) and printed out in the format shown
     0110    450 ;        below under  output parameters: .  The queue is accessed until
     0110    451 ;        empty.
     0110    452 ;
     0110    453 ; Calling sequence:
     0110    454 ;
     0110    455 ;        CALL     SUMMARY
     0110    456 ;
     0110    457 ; Input parameters:      None
     0110    458 ;
     0110    459 ; Implicit inputs:
     0110    460 ;
     0110    461 ;        The test summary information block has been dynamically filled at
     0110    462 ;        run time.
     0110    463 ;
     0110    464 ; Output parameters:
     0110    465 ;
     0110    466 ;        Any tests having errors will have information printed out in the
     0110    467 ;        following format:
     0110    468 ;*
     0110    469 ;        Test xx had yy hard errors; hh:mm:ss run time.
     0110    470 ;
     0110    471 ;        Where,
     0110    472 ;
     0110    473 ;        xx       represents the relevent test number.
     0110    474 ;        yy       represents the accumulated number of hard errors
     0110    475 ;                  encountered in the course of fulfillment of the
     0110    476 ;                  pass requirements.
     0110    477 ;
     0110    478 ;        hh:mm:ss represents the accumulated run time encountered for the
     0110    479 ;                  test in the course of fulfillment of pass requirements.
     0110    480 ;
     0110    481 ;        Note!!!!! any test not encountering errors will not appear
     0110    482 ;        as part of the summary printout (i.e. if no errors existed
     0110    483 ;        in the execution of the exerciser there would be no summary
     0110    484 ;        printout).
     0110    485 ;*
     0110    486 ;
     0110    487 ; Implicit outputs:      None
     0110    488 ;
     0110    489 ; Completion codes:      None
     0110    490 ;
     0110    491 ; Side effects:          None
     0110    492 ;
     0110    493 ;--
```

```
                                     0110   495         $DS_BGNSUMMARY  <R2, R3>
                                     0110                       .SAVE
                                 00000000                       .PSECT  SUMMARY,LONG
                                     0000       SUMMARY:
                               000C  0000                       .WORD   ^M<R2, R3>              ; ENTRY MASK
              52   0000000C'EF   D0  0002   496         MOVL    SUM_A_BLOCK, R2                 ; Get pointer to summary info.
                       53   01   D0  0009   497         MCVL    #1, R3                          ; Test number.
                                     000C   498 1$:                                     ; DO
                            62   D5  000C   499         TSTL    SUM_L_HARDCNT(R2)               ; Check for any errors and
                            2B   13  000E   500         BEQL    2$                              ;  skip report if none.
                                     0010   501         $ASCTIM_S -                             ; Convert run time to ASCII:
                                     0010   502                 TIMBUF=QWD_TIME, -              ;  QWD of buffer for ASCII time;
                                     0010   503                 TIMADR=SUM_Q_RUNTIME(R2), -     ;  64-bit run time;
                                     0010   504                 CVTFLG=#1                       ;  hh:mm:ss only flag.
                            01   DD  0010                       PUSHL   #1
                         04 A2   7F  0012                       PUSHAQ  SUM_Q_RUNTIME(R2)
                   0000008C'EF   7F  0015                       PUSHAQ  QWD_TIME
                            00   DD  001B                       PUSHL   #0
              00010248'GF   04   FB  001D                       CALLS   #4,G^SYS$ASCTIM
                                     0024   505         $DS_PRINTS_S    FMT_SUMMARY, -          ; Display summary information:
                                     0024   506                 R3, -                           ;  Test number;
                                     0024   507                 SUM_L_HARDCNT(R2), -            ;  Number of hard errors;
                                     0024   508                 #QWD_TIME                       ;  Test run time.
                   0000008C'8F   DD  0024                       PUSHL   #QWD_TIME
                            62   DD  002A                       PUSHL   SUM_L_HARDCNT(R2)
                            53   DD  002C                       PUSHL   R3
                   0000009F'EF   9F  002E                       PUSHAB  FMT_SUMMARY
              000100F8 9F   04   FB  0034                       CALLS   #$$N, @#DS$PRINTS
                       52   0C   C0  003B   509 2$:     ADDL2   #SUM_K_SIZE, R2                 ; Update block pointer to next
                                     003E   510                                                 ;  test.
           C6 53   00000054'FF   F3  003E   511         AOBLEQ  @L$A_TSTCNT, R3, 1$     ;   WHILE test # LEQ last test.
                                     0046   512                                         ; ENDDO
                                     0046   513         $DS_ENDSUMMARY
                                     0046       SUMMARY_X:
              00010058 9F   6E   FA  0046                       CALLG   (SP), @#DS$BREAK
                                 04  004D                       RET                     ; RETURN TO COMMAND MODE
                                 00000110                       .RESTORE
```

```
0110            .SBTTL  Summary Information Block Initialization Subroutine
0110    ;++
0110    ; Functional description:
0110    ;
0110    ;       This subroutine is responsible for building the structure for the
0110    ;       exerciser test summary information block.  This block will contain
0110    ;       the number of hard errors and run time for each test, which will be
0110    ;       displayed when a SUMMARY is requested.
0110    ;
0110    ; Calling sequence:
0110    ;
0110    ;       JSB     SUMMARY_INIT
0110    ;
0110    ; Input parameters:     None
0110    ;
0110    ; Implicit inputs:
0110    ;
0110    ;       L$A_TSTCNT      = Pointer to the number of tests in this program;
0110    ;       SUM_K_SIZE      = Number of bytes to be reserved for each test's data.
0110    ;
0110    ; Output parameters:    None
0110    ;
0110    ; Implicit outputs:
0110    ;
0110    ;       SUM_A_BLOCK     = Pointer to the memory reserved for the summary data;
0110    ;       @SUM_A_BLOCK    = The summary data block itself is cleared.
0110    ;
0110    ; Completion codes:     None
0110    ;
0110    ; Side effects:         None
0110    ;
0110    ;--
```

```
                                 0110     548 SUMMARY_INIT:
                       0C    BB  0110     549         PUSHR    #^M<R2, R3>
     52   0C    00000054'FF  C5  0112     550         MULL3    @L$A_TSTCNT, -            ; Compute the amount of buffer required
                                 011A     551                   #SUM_K_SIZE, R2         ;  for the summary information block.
     53   52    00000200 8F  C7  011A     552         DIVL3    #512, R2, R3              ; Convert to pages,
                       53    D6  0122     553         INCL     R3                        ;  and round up.
                                 0124     554         $DS_GETBUF_S  R3, SUM_A_BLOCK    ; Request the buffer from the Supervisor.
                       00    DD  0124                          PUSHL    #0
                       00    DD  0126                          PUSHL    #0
                  FEE0 CF    7F  0128                          PUSHAQ   SUM_A_BLOCK
                       53    DD  012C                          PUSHL    R3
          00010120 9F  04    FB  012E                          CALLS    #4, @#DS$GETBUF
                    0F 50    E8  0135     555         BLBS     R0,10$                    ; Branch if no error
                                 0138     556         $DS_ERRSYS_S
                       00    DD  0138                          PUSHL    #0
                       00    DD  013A                          PUSHL    #0
                       00    DD  013C                          PUSHL    #0
                       01    DD  013E                          PUSHL    #$ER
                                 0140         ;;; GLOBAL ERROR 1
          000100C0 9F  04    FB  0140                          CALLS    #$$M, @#DS$ERRSYS
                                 0147     557 10$:
             FEC8 CF   53    D0  0147     558         MOVL     R3, SUM_BLK_SIZE          ; Save the page court.
             52    FEBC CF   C0  014C     559         ADDL2    SUM_A_BLOCK, R2           ; Compute end of block address
        FEB9 CF   52   01    C3  0151     560         SUBL3    #1, R2, SUM_A_BLOCK+4     ;  and save it.
                       72    D4  0157     561 20$:    CLRL     -(R2)                     ; Clear out
             FEAE CF   52    D1  0159     562         CMPL     R2, SUM_A_BLOCK           ;  the summary information
                       F7    14  015E     563         BGTR     20$                       ;  block.
                                 0160     564
                       0C    BA  0160     565         POPR     #^M<R2, R3>
                             05  0162     566         RSB                                ; Return to mainline code.
```

```
0163            .SBTTL  Summary Information Input Subroutine
0163      ;++
0163      ; Functional description:
0163      ;
0163      ;         This subroutine is responsible for dynamically filling the test
0163      ;         summary information block with error data at the end of each test.
0163      ;
0163      ; Calling sequence:
0163      ;
0163      ;         JSB PART_INIT
0163      ;
0163      ; Input parameters:     None
0163      ;
0163      ;
0163      ; Implicit inputs:
0163      ;
0163      ;         DSA$GL_TESTNO = Current test number;
0163      ;         HARDCNT =       Number of hard errors, this test, this pass;
0163      ;         Q_TEST_TIME =   Start time of test;
0163      ;         SUM_A_BLOCK =   Base address of the summary information block;
0163      ;         SUM_K_SIZE =    Size of an entry in the summary information block.
0163      ;
0163      ; Output parameters:    None
0163      ;
0163      ; Implicit outputs:
0163      ;
0163      ;         SUM_L_HARDCNT(R2)       = Number of accumulated hard errors;
0163      ;         SUM_Q_RUNTIME(R2)       = Accumulated test run time.
0163      ;
0163      ; Completion codes:     None
0163      ;
0163      ; Side effects:         None
0163      ;
0163      ;--
```

```
                                     0163     603 PART_INIT::
          52   0C    0000FE50'EF  C5 0163     604         MULL3   DSA$GL_TESTNO, -           ; Grab the current test number and
                                     016B     605                  #SUM_K_SIZE, R2           ;  compute displacement into the
                   52   FE9D CF  C0  016B     606         ADDL2   SUM_A_BLOCK, R2           ;  summary information block.
                   62   FE8C CF  C0  0170     607         ADDL2   HARDCNT, -                ; Update the tests hard count total.
                                     0175     608                  SUM_L_HARDCNT(R2)
                        FE87 CF  D4  0175     609         CLRL    HARDCNT                   ; Reinitialize exerciser hard
                                     0179     610                                           ;  error count for next test.
                                     0179     611 ;
                                     0179     612 ; Compute current run time and add it to previous passes' run time.
                                     0179     613 ;
              04 A2     FE87 CF  C0  0179     614         ADDL2   Q_TEST_TIME, -            ; Factor in absolute start time.
                                     017F     615                  SUM_Q_RUNTIME(R2)
              04 A2     FE81 CF  D8  017F     616         ADWC    Q_TEST_TIME, SUM_Q_RUNTIME(R2)
                                     0185     617         $GETTIM_S       Q_TEST_TIME       ; Get current time.
                        FE7B CF  7F  0185                         PUSHAQ  Q_TEST_TIME
             00010378'GF    01   FB  0189                         CALLS   #1,G^SYS$GETTIM
              04 A2     FE70 CF  C2  0190     618         SUBL2   Q_TEST_TIME, -            ; Factor out absolute time;
                                     0196     619                  SUM_Q_RUNTIME(R2)        ;  result is delta time.
              04 A2     FE6A CF  D9  0196     620         SBWC    Q_TEST_TIME, SUM_Q_RUNTIME(R2)
                                     019C     621
                                 05  019C     622         RSB                               ;Back into the stream of things!
                                     019D     623
                                     019D     624         $DS_ENDMOD
                                     019D     625         .END
```

```
$$E                            = 00000001                DS$FREEDBGSYM                   000101B8
$$M                            = 00000004                DS$GETBUF                       00010120
$$N                            = 00000000                DS$GETMEM                       00010130
$$S                            = FFFFFFFF                DS$GETTERM                      000101C8
$ENV                           = 00000001  G             DS$GPHARD                       00010018
$ER                            = 00000002                DS$HELP                         000101B0
$MO                            = 00000001  G             DS$INITSCB                      00010170
$ST                            = 00000000                DS$INLOOP                       00010048
$TN                            = 00000000                DS$K_ERROR                    = 00000002
AEM                              00000128 R      01      DS$K_NORMAL                   = 00000001
AKM                              00000121 R      01      DS$K_SEVERE                   = 00000004
ALL                            = 00000009  G             DS$K_SUBSYS                   = 00000066
AL_DEVTYP                        00000084 R      08      DS$K_WARNING                  = 00000000
ASM                              00000132 R      01      DS$LOAD                         00010198
AUM                              0000013D R      01      DS$LOADPCS                      000101C0
A_BUFFER_BEGIN                   00000000 RG     01      DS$MAPDBGBLOCK                  00010118
A_BUFFER_END                     00000004 RG     01      DS$MEMSIZE                      000101D8
A_HALT_BUFFER                    00000010 RG     01      DS$MMOFF                        00010158
A_HEADEND                        00000058 R      03      DS$MMON                         00010150
A_MCHK_BUFFER                    00000018 RG     01      DS$MOVPHY                       00010148
A_PSLMNE                         000000CC RG     01      DS$MOVVRT                       00010140
A_PWR_BUFFER                     00000008 RG     01      DS$PARSE                        000100B8
A_WCS_BUFFER                     00000020 RG     01      DS$PRINTB                       000100E0
BUF_SIZE                       = 0000000B                DS$PRINTF                       000100F0
CEP_FUNCTIONAL                 = 00000000                DS$PRINTS                       000100F8
CEP_REPAIR                     = 00000001                DS$PRINTSIG                     00010100
CLEAN_UP                         00000000 R      0A      DS$PRINTX                       000100E8
CLEAN_UP_X                       000000DE R      0A      DS$PROBE                        000101A0
DEFAULT                        = 00000000  G             DS$RELBUF                       00010128
DEV REG                          00000000 R      08      DS$RELMEM                       00010138
DISPATCH                         00000000 R      06      DS$SERVICE_DISPATCHER           000101D0
DS$ABORT                         00010020                DS$SETIPL                       00010178
DS$ASKADR                        00010090                DS$SETMAP                       00010188
DS$ASKDATA                       00010080                DS$SETPRIEXV                    00010110
DS$ASKLGCL                       00010098                DS$SETVEC                       00010160
DS$ASKSTR                        000100A0                DS$SHOCHAN                      00010190
DS$ASKVLD                        00010088                DS$SUMMARY                      00010028
DS$ATTACH                        000101A8                DS$WAITMS                       00010060
DS$BGNSUB                        00010030                DS$WAITUS                       00010068
DS$BRANCH                        000100A8                DS$_ARITH                     = 006600D0
DS$BREAK                         00010058                DS$_ASBE                      = 00660118
DS$CANWAIT                       00010070                DS$_BADLINK                   = 006600F0
DS$CHANNEL                       00010180                DS$_BADTYPE                   = 006600E8
DS$CKLOOP                        00010040                DS$_BIIC                      = 00660120
DS$CLRVEC                        00010168                DS$_CHME                      = 006600A8
DS$CNTRLC                        00010078                DS$_CHMK                      = 006600E0
DS$CVTREG                        000100B0                DS$_DEVNAME                   = 00660108
DS$ENDPASS                       00010010                DS$_ERROR                     = 00660002
DS$ENDSUB                        00010038                DS$_FHWE                      = 00660068
DS$ERRDEV                        000100C8                DS$_FRAGBUF                   = 00660080
DS$ERRHARD                       000100D0                DS$_ICBUSY                    = 006600C8
DS$ERRPREP                       00010108                DS$_ICERR                     = 006600C0
DS$ERRSOFT                       000100D8                DS$_IHWE                      = 00660060
DS$ERRSYS                        000100C0                DS$_ILLCHAR                   = 00660018
DS$ESCAPE                        00010050                DS$_ILLPAGCNT                 = 00660078
```

```
DS$_ILLUNIT                  = 00660100                ENV$_REPAIR                    = 00000001
DS$_INSFMEM                  = 00660050                ENV$_SUPER                     = 00000001
DS$_IPL2HI                   = 006600B8                ENV$_SYSTEM                    = 00000001
DS$_IVADDR                   = 00660040                FMT_SUMMARY                      0000009F R       08
DS$_IVVECT                   = 00660038                GPR_TABLE                        00000048 RG      01
DS$_KRNLSTK                  = 00660090                HALT                           = 00000004  G
DS$_LOGIC                    = 00660070                HARDCNT                          00000000 RG      08
DS$_MCHK                     = 00660088                HP$A_DEPENDENT                   00000032
DS$_MMOFF                    = 00660058                HP$A_DEVICE                      00000018
DS$_NEEDUNIT                 = 006600F8                HP$A_DVA                         0000001C
DS$_NODE                     = 00660128                HP$A_LINK                        00000020
DS$_NOPCS                    = 00660110                HP$B_DRIVE                       0000000B
DS$_NORMAL                   = 00660001                HP$B_FLAGS                       0000000A
DS$_NOSUPPORT                = 006600B1                HP$Q_DEVICE                      00000000
DS$_NOTDON                   = 00660030                HP$T_DEVICE                      0000000C
DS$_NOTIMP                   = 006600B0                HP$T_TYPE                        00000026
DS$_NULLSTR                  = 00660010                HP$W_SIZE                        00000008
DS$_OVERFLOW                 = 00660008                HP$W_VECTOR                      00000024
DS$_POWER                    = 00660098                INITIALIZE                       00000000 R       09
DS$_PROGERR                  = 00660020                INITIALIZE_X                     000000A1 R       09
DS$_SEVERE                   = 00660004                IPR                            = 00000006  G
DS$_TRANSL                   = 006600A0                KA$B_ACC_TYPE                    00000042
DS$_TRUNCATE                 = 00660028                KA$B_SID_TYPE                    0000003D
DS$_UNEXPINT                 = 006600D8                KA$K_LEN                         00000043
DS$_VASFULL                  = 00660048                KA$L_FLAGS                       00000032
DS$_WARNING                  = 00660000                KA$L_SID                         0000003A
DSA$AL_APTMAIL                 0000FE00                KA$W_LATENCY                     0000003E
DSA$AT_APTTXT                  0000FA00                KA$W_PROGRESS                    00000040
DSA$GL_APTCOM                  0000FE04                KA$W_RESERVED                    00000038
DSA$GL_DEVLEN                  0000FE58                KA$W_WCS                         00000036
DSA$GL_ERRNO                   0000FE44                KA750                          = 00000001
DSA$GL_EVENT                   0000FE48                KA_PTABLE                        00000142 RG      01
DSA$GL_FLAGS                   0000FE00                L$A_CCP                          00000040 R       03
DSA$GL_MSGTYP                  0000FE40                L$A_DEVP                         0000001C R       03
DSA$GL_PASSES                  0000FE08                L$A_DREG                         00000024 R       03
DSA$GL_PASSNO                  0000FE54                L$A_DTP                          00000018 R       03
DSA$GL_SECTNO                  0000FE10                L$A_ICP                          0000003C R       03
DSA$GL_SID                     0000FE14                L$A_LASTAD                       00000014 R       03
DSA$GL_SUBTNO                  0000FE4C                L$A_NAME                         00000008 R       03
DSA$GL_TESTNO                  0000FE50                L$A_REPP                         00000044 R       03
DSA$GL_UNITS                   0000FE0C                L$A_SECNAM                       00000050 R       03
DSA$GQ_MSGPTR                  0000FE68                L$A_STATAB                       00000048 R       03
DSA$GT_DEVNAM                  0000FE5C                L$A_TSTCNT                       00000054 R       03
DSA$V_PASS0                  = 0000001D                L$L_ENVIRON                      00000004 R       03
ENV$M_DOMAIN                 = 00000002                L$L_ERRTYP                       0000004C R       03
ENV$M_LEVEL                  = 00000001                L$L_HEADLENGTH                   00000000 R       03
ENV$M_SUPER                  = 000003FC                L$L_REV                          0000000C R       03
ENV$S_DOMAIN                 = 00000001                L$L_UNIT                         00000020 R       03
ENV$S_LEVEL                  = 00000001                L$L_UPDATE                       00000010 R       03
ENV$S_SUPER                  = 00000008                LASTAD                           00000000 R       04
ENV$V_DOMAIN                 = 00000001                L_ACT_ISP                        00000038 RG      01
ENV$V_LEVEL                  = 00000000                L_ACT_KSP                        00000034 RG      01
ENV$V_SUPER                  = 00000002                L_ACT_PSL                        0000003C RG      01
ENV$_CPU                     = 00000000                L_ACT_SP                         00000030 RG      01
ENV$_FUNCTIONAL              = 00000000                L_EXP_AP                         00000078 RG      01
```

```
L_EXP_FP                         0000007C RG    01      SCB$L_CHMS                         00000048
L_EXP_ISP                        00000088 RG    01      SCB$L_CHMU                         0000004C
L_EXP_KSP                        00000084 RG    01      SCB$L_COMPAT                       00000030
L_EXP_PSL                        0000008C RG    01      SCB$L_KNLSTK                       00000008
L_EXP_SP                         00000080 RG    01      SCB$L_MACHCHK                      00000004
L_ISP                            00000044 RG    01      SCB$L_OPCCUS                       00000014
L_KSP                            00000040 RG    01      SCB$L_OPCDEC                       00000010
L_REV_CHECK                      00000090 RG    01      SCB$L_POWER                        0000000C
L_TSTCNT                         00000000 R     05      SCB$L_RADRMOD                      0000001C
MANUAL                         = 00000001  G            SCB$L_ROPRAND                      00000018
MCHK                           = 00000005  G            SCB$L_RXDB                         000000F8
MODELIST                         0000010D RG    01      SCB$L_SFTLVL1                      00000084
NOT_PCS                          0000001A R     0A      SCB$L_SFTLVL10                     000000A8
PAR$M_ATDEF                    = 00000010               SCB$L_SFTLVL11                     000000AC
PAR$M_ATHI                     = 00000008               SCB$L_SFTLVL12                     000000B0
PAR$M_ATLO                     = 00000004               SCB$L_SFTLVL13                     000000B4
PAR$M_NODEF                    = 00000002               SCB$L_SFTLVL14                     000000B8
PAR$V_ATDEF                    = 00000004               SCB$L_SFTLVL15                     000000BC
PAR$V_ATHI                     = 00000003               SCB$L_SFTLVL2                      00000088
PAR$V_ATLO                     = 00000002               SCB$L_SFTLVL3                      0000008C
PAR$V_NODEF                    = 00000001               SCB$L_SFTLVL4                      00000090
PAR$_BIN                       = 00000002               SCB$L_SFTLVL5                      00000094
PAR$_DEC                       = 0000000A               SCB$L_SFTLVL6                      00000098
PAR$_HEX                       = 00000010               SCB$L_SFTLVL7                      0000009C
PAR$_NO                        = 00000000               SCB$L_SFTLVL8                      000000A0
PAR$_OCT                       = 00000008               SCB$L_SFTLVL9                      000000A4
PAR$_YES                       = 00000001               SCB$L_TBIT                         00000028
PART_INIT                        00000163 RG    08      SCB$L_TIMER                        000000C0
PCS                            = 00000008  G            SCB$L_TRANSL                       00000024
PCS$A_PCS_BUFFER                 00000028 RG    01      SCB$L_TXDB                         000000FC
POWER_FAIL                     = 00000003  G            SCB$L_ZERO                         00000000
PR$_SID                        = 0000003E               SECTION                            00000051 R     08
PR750$_CADR                    = 00000025               SEP_FUNCTIONAL                   = 00000002
PR750$_CSRS                    = 0000001C               SEP_REPAIR                       = 00000003
PR750$_CSTS                    = 0000001E               SIZ...                           = 00000001
PSL$K_EXEC                     = 00000001               SPACE1                             000000CF RG    08
PSL$K_KERNEL                   = 00000000               SPACE10                            00000105 RG    08
PSL$K_SUPER                    = 00000002               SPACE2                             000000D1 RG    08
PSL$K_USER                     = 00000003               SPACE3                             000000D4 RG    08
PSL$M_CBIT                     = 00000001               SPACE4                             000000D8 RG    08
PSL$M_NBIT                     = 00000008               SPACE5                             000000DD RG    08
PSL$M_VBIT                     = 00000002               SPACE6                             000000E3 RG    08
PSL$M_ZBIT                     = 00000004               SPACE7                             000000EA RG    08
PSL$V_CBIT                     = 00000000               SPACE8                             000000F2 RG    08
PSL$V_NBIT                     = 00000003               SPACE9                             000000FB RG    08
PSL$V_VBIT                     = 00000001               SUMMARY                            00000000 R     0B
PSL$V_ZBIT                     = 00000002               SUMMARY_INIT                       00000110 R     08
QWD_TIME                         0000008C R     08      SUMMARY_X                          00000046 R     0B
Q_TEST_TIME                      00000004 R     08      SUM_A_BLOCK                        0000000C R     08
REG_TABLE                        00000094 RG    01      SUM_BLK_SIZE                       00000014 R     08
SCB$L_ACCESS                     00000020               SUM_K_SIZE                         0000000C
SCB$L_ARITH                      00000034               SUM_L_HARDCNT                      00000000
SCB$L_BREAK                      0000002C               SUM_Q_RUNTIME                      00000004
SCB$L_CHME                       00000044               SYS$ALLOC                          00010238
SCB$L_CHMK                       00000040               SYS$ASCTIM                         00010248  G
```

```
SYS$ASSIGN                        00010250
SYS$BINTIM                        00010258
SYS$CANCEL                        00010260
SYS$CANTIM                        00010268
SYS$CLOSE                         000105B8
SYS$CLREF                         00010298
SYS$CONNECT                       000105C0
SYS$DALLOC                        000102D8
SYS$DASSGN                        000102E0
SYS$DISCONNECT                    000105D0
SYS$FAO                           00010350
SYS$FAOL                          00010358
SYS$GET                           00010580
SYS$GETCHN                        000104C8
SYS$GETTIM                        00010378  G
SYS$LKWSET                        000103A0
SYS$NUMTIM                        000103B8
SYS$OPEN                          00010608
SYS$QIO                           000103C8
SYS$QIOW                          00010200
SYS$READ                          00010590
SYS$READEF                        000103D0
SYS$SETAST                        000103F8
SYS$SETEF                         00010400
SYS$SETIMR                        00010420
SYS$SETPRI                        00010428
SYS$SETPRT                        00010430
SYS$SETRWM                        00010438
SYS$SETSWM                        00010448
SYS$ULKPAG                        00010460
SYS$ULWSET                        00010468
SYS$UNWIND                        00010470
SYS$WAITFR                        00010478
SYS$WFLAND                        00010488
SYS$WFLOR                         00010490
S_ALL                             0000004D R      08
S_DEFAULT                         00000018 R      08
S_HALT                            00000037 R      08
S_IPR                             00000041 R      08
S_MANUAL                          00000020 R      08
S_MCHK                            0000003C R      08
S_PCS                             00000049 R      08
S_POWER_FAIL                      0000002C R      08
S_TU58                            00000027 R      08
S_WCS                             00000045 R      08
TU58                            = 00000002  G
T_KA750                           0000007D R      08
T_NAME                            00000058 R      03
T_TIME                            00000094 R      08
WCS                             = 00000007  G
```

```
                                                +-----------------+
                                                ! Psect synopsis !
                                                +-----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
ECKAX1                          00000146  (   326.)  01 (  1.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD  NOWRT NOVEC PAGE
$ABS$                           00010610  (67088.)  02 (  2.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
$HEADER                         00000092  (   146.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD  NOWRT NOVEC PAGE
_LAST                           00000000  (     0.)  04 (  4.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC PAGE
$TSTCNT                         00000000  (     0.)  05 (  5.)  NOPIC   USR   OVR   REL   LCL NOSHR NOEXE   RD  NOWRT NOVEC LONG
DISPATCH                        00000000  (     0.)  06 (  6.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD  NOWRT NOVEC LONG
DISPATCH_X                      00000018  (    24.)  07 (  7.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD  NOWRT NOVEC LONG
DATA                            0000019D  (   413.)  08 (  8.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC PAGE
INITIALIZE                      000000A9  (   169.)  09 (  9.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC LONG
CLEANUP                         000000E6  (   230.)  0A ( 10.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC LONG
SUMMARY                         0000004E  (    78.)  0B ( 11.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC LONG
```

```
                                             +-------------------------+
                                             ! Symbol Cross Reference !
                                             +-------------------------+

SYMBOL                          VALUE        DEFINITION      REFERENCES...
------                          -----        ----------      -------------
$$E                            =00000001    411    (8)      373    (7)      440    (8)
$$M                            =00000004    556    (12)     556    (12)
$$N                            =00000000    556    (12)     273    (5)      283    (5)      508    (10)     556    (12)
$$S                            =FFFFFFFF    67     (2)
$ENV                           =00000001    67     (2)      183    (3)
$ER                            =00000002    556    (12)     356    (7)      411    (8)    #-556    (12)
$MO                            =00000001    624    (14)     624    (14)
$ST                            =00000000    440    (8)      373    (7)      440    (8)      556    (12)
$TN                            =00000000    67     (2)      356    (7)      411    (8)      495    (10)     624    (14)
AEM                             00000128-R  163    (2)      158    (2)
AKM                             00000121-R  162    (2)      157    (2)
ALL                            =00000009    273    (5)
AL_DEVTYP                       00000084-R  283    (5)      183    (3)
ASM                             00000132-R  164    (2)      159    (2)
AUM                             0000013D-R  165    (2)      160    (2)
A_BUFFER_BEGIN                  00000000-R  92     (2)      365    (7)
A_BUFFER_END                    00000004-R  93     (2)
A_HALT_BUFFER                   00000010-R  95     (2)      367    (7)
A_HEADEND                       00000058-R  183    (3)      183    (3)
A_MCHK_BUFFER                   00000018-R  96     (2)      368    (7)
A_PSLMNE                        000000CC-R  152    (2)
A_PWR_BUFFER                    00000008-R  94     (2)      366    (7)
A_WCS_BUFFER                    00000020-R  97     (2)      369    (7)
BIT...                         =00000011    83     (2)      67     (2)      79     (2)      82     (2)      83     (2)
BUF_SIZE                       =0000000B    294    (5)      296    (5)      299    (5)
CEP_FUNCTIONAL                 =00000000    67     (2)
CEP_REPAIR                     =00000001    67     (2)      67     (2)
CLEAN_UP                        00000000-R  411    (8)      183    (3)
CLEAN_UP_X                      000000DE-R  440    (8)
DEFAULT                        =00000000    273    (5)
DEV_REG                         00000000-R  221    (4)      183    (3)
DISPATCH                        00000000-R  196    (3)      183    (3)
DS$BREAK                        00010058                    373    (7)      440    (8)      513    (10)
DS$CLRVEC                       00010168                    420    (8)      421    (8)      422    (8)      423    (8)
                                                            424    (8)      425    (8)      426    (8)      427    (8)
                                                            428    (8)      429    (8)      430    (8)
DS$ENDPASS                      00010010                    372    (7)
DS$ERRSYS                       000100C0                    556    (12)
DS$GETBUF                       00010120                    365    (7)      366    (7)      367    (7)      368    (7)
                                                            369    (7)      370    (7)      554    (12)
DS$GPHARD                       00010018                    362    (7)
DS$INITSCB                      00010170                    432    (8)
DS$K_ERROR                     =00000002    79     (2)
DS$K_NORMAL                    =00000001    79     (2)
DS$K_SEVERE                    =00000004    79     (2)
DS$K_SUBSYS                    =00000066    79     (2)      79     (2)
DS$K_WARNING                   =00000000    79     (2)
DS$LOADPCS                      000101C0                    417    (8)
```

```
DS$MMOFF                        00010158                        431     (8)
DS$PRINTS                       000100F8                        508     (10)
DS$RELBUF                       00010128                        434     (8)        435     (8)
DS$SETIPL                       00010178                        439     (8)
DS$SUMMARY                      00010028                        433     (8)
DS$_ARITH                      =006600D0       79      (2)
DS$_ASBE                       =00660118       79      (2)
DS$_BADLINK                    =006600F0       79      (2)
DS$_BADTYPE                    =006600E8       79      (2)
DS$_BIIC                       =00660120       79      (2)
DS$_CHME                       =006600A8       79      (2)
DS$_CHMK                       =006600E0       79      (2)
DS$_DEVNAME                    =00660108       79      (2)
DS$_ERROR                      =00660002       79      (2)
DS$_FHWE                       =00660068       79      (2)
DS$_FRAGBUF                    =00660080       79      (2)
DS$_ICBUSY                     =006600C8       79      (2)
DS$_ICERR                      =006600C0       79      (2)
DS$_IHWE                       =00660060       79      (2)
DS$_ILLCHAR                    =00660018       79      (2)
DS$_ILLPAGCNT                  =00660078       79      (2)
DS$_ILLUNIT                    =00660100       79      (2)
DS$_INSFMEM                    =00660050       79      (2)
DS$_IPL2HI                     =006600B8       79      (2)
DS$_IVADDR                     =00660040       79      (2)
DS$_IVVECT                     =00660038       79      (2)
DS$_KRNLSTK                    =00660090       79      (2)
DS$_LOGIC                      =00660070       79      (2)
DS$_MCHK                       =00660088       79      (2)
DS$_MMOFF                      =00660058       79      (2)
DS$_NEEDUNIT                   =006600F8       79      (2)
DS$_NODE                       =00660128       79      (2)
DS$_NOPCS                      =00660110       79      (2)
DS$_NORMAL                     =00660001       79      (2)
DS$_NOSUPPORT                  =0066C0B1       79      (2)
DS$_NOTDON                     =00660030       79      (2)
DS$_NOTIMP                     =006600B0       79      (2)        79      (2)
DS$_NULLSTR                    =00660010       79      (2)
DS$_OVERFLOW                   =00660008       79      (2)
DS$_POWER                      =00660098       79      (2)
DS$_PROGERR                    =00660020       79      (2)
DS$_SEVERE                     =00660004       79      (2)
DS$_TRANSL                     =006600A0       79      (2)
DS$_TRUNCATE                   =00660028       79      (2)
DS$_UNEXPINT                   =006600D8       79      (2)
DS$_VASFULL                    =00660048       79      (2)
DS$_WARNING                    =00660000       79      (2)        79      (2)
DSA$GL_FLAGS                    0000FE00                        359     (7)
DSA$GL_TESTNO                   0000FE50                      #-604     (14)
DSA$V_PASS0                    =0000001D                      #-359     (7)
ENV$M_DOMAIN                   =00000002       67      (2)
ENV$M_LEVEL                    =00000001       67      (2)
ENV$M_SUPER                    =000003FC       67      (2)
ENV$S_DOMAIN                   =00000001       67      (2)
ENV$S_LEVEL                    =00000001       67      (2)
```

```
ENV$S_SUPER                      =00000008      67      (2)
ENV$V_DOMAIN                     =00000001      67      (2)      67      (2)
ENV$V_LEVEL                      =00000000      67      (2)      67      (2)
ENV$V_SUPER                      =00000002      67      (2)      183     (3)
ENV$_CPU                         =00000000      67      (2)      67      (2)
ENV$_FUNCTIONAL                  =00000000      67      (2)      67      (2)
ENV$_REPAIR                      =00000001      67      (2)      67      (2)
ENV$_SUPER                       =00000001      67      (2)      183     (3)
ENV$_SYSTEM                      =00000001      67      (2)      67      (2)
FMT_SUMMARY                       0000009F-R    306     (5)      508     (10)
GPR_TABLE                         00000048-R    107     (2)
HALT                             =00000004      273     (5)
HARDCNT                           00000000-R    227     (4)     *-357    (7)      *-607     (14)     *-609     (14)
INITIALIZE                        00000000-R    356     (7)      183     (3)
INITIALIZE_X                      000000A1-R    373     (7)
IPR                              =00000006      273     (5)
KA750                            =00000001      283     (5)
KA_PTABLE                         00000142-R    209     (3)      362     (7)
L$A_CCP                           00000040-R    183     (3)
L$A_DEVP                          0000001C-R    183     (3)
L$A_DREG                          00000024-R    183     (3)
L$A_DTP                           00000018-R    183     (3)
L$A_ICP                           0000003C-R    183     (3)
L$A_LASTAD                        00000014-R    183     (3)
L$A_NAME                          00000008-R    183     (3)
L$A_REPP                          00000044-R    183     (3)
L$A_SECNAM                        00000050-R    183     (3)
L$A_STATAB                        00000048-R    183     (3)
L$A_TSTCNT                        00000054-R    183     (3)     *-511    (10)     *-550     (12)
L$L_ENVIRON                       00000004-R    183     (3)
L$L_ERRTYP                        0000004C-R    183     (3)
L$L_HEADLENGTH                    00000000-R    183     (3)
L$L_REV                           0000000C-R    183     (3)
L$L_UNIT                          00000020-R    183     (3)
L$L_UPDATE                        00000010-R    183     (3)
LASTAD                            00000000-R    183     (3)      183     (3)
L_ACT_ISP                         00000038-R    102     (2)
L_ACT_KSP                         00000034-R    101     (2)
L_ACT_PSL                         0000003C-R    103     (2)
L_ACT_SP                          00000030-R    100     (2)
L_EXP_AP                          00000078-R    120     (2)
L_EXP_FP                          0000007C-R    121     (2)
L_EXP_ISP                         00000088-R    124     (2)
L_EXP_KSP                         00000084-R    123     (2)
L_EXP_PSL                         0000008C-R    125     (2)
L_EXP_SP                          00000080-R    122     (2)
L_ISP                             00000044-R    105     (2)
L_KSP                             00000040-R    104     (2)
L_REV_CHECK                       00000090-R    127     (2)     *-414    (8)
L_TSTCNT                          00000000-R    183     (3)      183     (3)
MANUAL                           =00000001      273     (5)
MCHK                             =00000005      273     (5)
MODELIST                          0000010D-R    156     (2)
NOT_PCS                           0000001A-R    419     (8)     *-416    (8)
PAR$M_ATDEF                      =00000010      82      (2)
```

```
PAR$M_ATHI                       =00000008     82       (2)
PAR$M_ATLO                       =00000004     82       (2)
PAR$M_NODEF                      =00000002     82       (2)
PAR$V_ATDEF                      =00000004     82       (2)
PAR$V_ATHI                       =00000003     82       (2)
PAR$V_ATLO                       =00000002     82       (2)
PAR$V_NODEF                      =00000001     82       (2)
PAR$_BIN                         =00000002     82       (2)
PAR$_DEC                         =0000000A     82       (2)
PAR$_HEX                         =00000010     82       (2)
PAR$_NO                          =00000000     82       (2)
PAR$_OCT                         =00000008     82       (2)
PAR$_YES                         =00000001     82       (2)
PART_INIT                         00000163-R   603      (14)
PCS                              =00000008     273      (5)
PCS$A_PCS_BUFFER                  00000028-R   98       (2)       370      (7)
POWER_FAIL                       =00000003     273      (5)
PR$_SID                          =0000003E                      #-412      (8)
PR750$_CADR                      =00000025                      #-438      (8)
PR750$_CSRS                      =0000001C                      #-436      (8)
PR750$_CSTS                      =0000001E                      #-437      (8)
PSL$K_EXEC                       =00000001     83       (2)
PSL$K_KERNEL                     =00000000     83       (2)
PSL$K_SUPER                      =00000002     83       (2)
PSL$K_USER                       =00000003     83       (2)
PSL$M_CBIT                       =00000001     83       (2)
PSL$M_NBIT                       =00000008     83       (2)
PSL$M_VBIT                       =00000002     83       (2)
PSL$M_ZBIT                       =00000004     83       (2)
PSL$V_CBIT                       =00000000     83       (2)
PSL$V_NBIT                       =00000003     83       (2)
PSL$V_VBIT                       =00000001     83       (2)
PSL$V_ZBIT                       =00000002     83       (2)
QWD_TIME                          0000008C-R   295      (5)       504      (10)     #-508      (10)
Q_TEST_TIME                       00000004-R   229      (4)     #-614      (14)     #-616      (14)       617      (14)     #-618      (14)
                                                                #-620      (14)
REG_TABLE                         00000094-R   132      (2)
SCB$L_CHME                        00000044                      #-422      (8)
SCB$L_CHMK                        00000040                      #-425      (8)
SCB$L_CHMS                        00000048                      #-423      (8)
SCB$L_CHMU                        0000004C                      #-424      (8)
SCB$L_MACHCHK                     00000004                      #-426      (8)
SCB$L_OPCCUS                      00000014                      #-427      (8)
SCB$L_OPCDEC                      00000010                      #-428      (8)
SCB$L_POWER                       0000000C                      #-421      (8)
SCB$L_RXDB                        000000F8                      #-430      (8)
SCB$L_TIMER                       000000C0                      #-420      (8)
SCB$L_TXDB                        000000FC                      #-429      (8)
SECTION                           00000051-R   273      (5)       183      (3)
SEP_FUNCTIONAL                   =00000002     67       (2)
SEP_REPAIR                       =00000003     67       (2)
SIZ...                           =00000001     83       (2)        67      (2)        82      (2)        83      (2)
SPACE1                            000000CF-R   313      (5)
SPACE10                           00000105-R   322      (5)
SPACE2                            000000D1-R   314      (5)
```

```
SPACE3                          000000D4-R    315     (5)
SPACE4                          000000D8-R    316     (5)
SPACE5                          000000DD-R    317     (5)
SPACE6                          000000E3-R    318     (5)
SPACE7                          000000EA-R    319     (5)
SPACE8                          000000F2-R    320     (5)
SPACE9                          000000FB-R    321     (5)
SUMMARY                         00000000-R    495     (10)      183     (3)
SUMMARY_INIT                    00000110-R    548     (12)      363     (7)
SUMMARY_X                       00000046-R    513     (10)
SUM_A_BLOCK                     0000000C-R    245     (4)     #-496     (10)      554     (12)    #-559     (12)    #-560     (12)
                                                              #-562     (12)    #-606     (14)
SUM_BLK_SIZE                    00000014-R    257     (4)     #-435     (8)     #-558     (12)
SUM_K_SIZE                      0000000C      253     (4)     #-509     (10)    #-551     (12)    #-605     (14)
SUM_L_HARDCNT                   00000000      251     (4)     #-499     (10)    #-508     (10)    #-608     (14)
SUM_Q_RUNTIME                   00000004      252     (4)       504     (10)    #-615     (14)    #-616     (14)    #-619     (14)
                                                              #-620     (14)
SYS$ASCTIM                      00010248                        504     (10)
SYS$GETTIM                      00010378                        617     (14)
S_ALL                           0000004D-R    273     (5)       273     (5)
S_DEFAULT                       00000018-R    273     (5)       273     (5)
S_HALT                          00000037-R    273     (5)       273     (5)
S_IPR                           00000041-R    273     (5)       273     (5)
S_MANUAL                        00000020-R    273     (5)       273     (5)
S_MCHK                          0000003C-R    273     (5)       273     (5)
S_PCS                           00000049-R    273     (5)       273     (5)
S_POWER_FAIL                    0000002C-R    273     (5)       273     (5)
S_TU58                          00000027-R    273     (5)       273     (5)
S_WCS                           00000045-R    273     (5)       273     (5)
TU58                           =00000002      273     (5)
T_KA750                         0000007D-R    283     (5)       283     (5)
T_NAME                          00000058-R    183     (3)       183     (3)
T_TIME                          00000094-R    299     (5)       297     (5)
WCS                            =00000007      273     (5)
```

```
                                                 +-------------------------+
                                                 ! Macros Cross Reference !
                                                 +-------------------------+

MACRO                SIZE           DEFINITION        REFERENCES...
-----                ----           ----------        -------------
$ASCTIM_S            1              502    (10)       502    (10)
$D1_ERR              1              556    (12)       556    (12)
$D1_ERR_S            1              556    (12)       556    (12)
$D1_PRINT_S          1              508    (10)       508    (10)
$D1_SEC              1              273    (5)        273    (5)
$DEF                 1              83     (2)        251    (4)        252    (4)        253    (4)
$DEFINI              1              77     (2)        77     (2)        78     (2)        80     (2)        81     (2)        83     (2)
                                                      84     (2)        85     (2)        86     (2)
$DS_BGNCLEAN         1              411    (8)        411    (8)
$DS_BGNINIT          1              356    (7)        356    (7)
$DS_BGNMOD           1              67     (2)        67     (2)
$DS_BGNSUMMARY       1              495    (10)       495    (10)
$DS_BPASS0           1              359    (7)        359    (7)
$DS_BREAK            1              373    (7)        373    (7)        440    (8)        513    (10)
$DS_CLRVEC_S         1              420    (8)        420    (8)        421    (8)        422    (8)        423    (8)        424    (8)
                                                      425    (8)        426    (8)        427    (8)        428    (8)        429    (8)
                                                      430    (8)
$DS_DEVTYP           1              283    (5)        283    (5)
$DS_DISPATCH         1              196    (3)        196    (3)
$DS_DSADEF           5              81     (2)        81     (2)
$DS_DSDEF            2              79     (2)        79     (2)
$DS_DSSDEF           8              80     (2)        80     (2)
$DS_ENDCLEAN         1              440    (8)        440    (8)
$DS_ENDINIT          1              373    (7)        373    (7)
$DS_ENDMOD           1              624    (14)       624    (14)
$DS_ENDPASS_G        1              372    (7)        372    (7)
$DS_ENDSUMMARY       1              513    (10)       513    (10)
$DS_ENVDEF           2              67     (2)        67     (2)
$DS_ERRSYS_S         1              556    (12)       556    (12)
$DS_GETBUF_S         1              365    (7)        365    (7)        366    (7)        367    (7)        368    (7)        369    (7)
                                                      370    (7)        554    (12)
$DS_GPHARD_S         1              362    (7)        362    (7)
$DS_HEADER           4              181    (3)        181    (3)
$DS_HPODEF           2              85     (2)        85     (2)
$DS_INITSCB_G        1              432    (8)        432    (8)
$DS_LOADPCS_S        1              417    (8)        417    (8)
$DS_MMOFF_G          1              431    (8)        431    (8)
$DS_PARDEF           1              82     (2)        82     (2)
$DS_PRINTS_S         1              505    (10)       505    (10)
$DS_PSLDEF           1              83     (2)        83     (2)
$DS_RELBUF_S         1              434    (8)        434    (8)        435    (8)
$DS_SCBDEF           3              78     (2)        78     (2)
$DS_SECTION          2              273    (5)        273    (5)
$DS_SETIPL_S         1              439    (8)        439    (8)
$DS_SUMMARY_G        1              433    (8)        433    (8)
$EQU                 1              83     (2)        67     (2)        79     (2)        82     (2)        83     (2)
$EQULS1              1              83     (2)        67     (2)        79     (2)        83     (2)
$EQULST              1              67     (2)        67     (2)        79     (2)        83     (2)
```

```
$GBLINI            2           67     (2)      67     (2)      79     (2)      82     (2)      83     (2)
$GETTIM_S          1           617    (14)     617    (14)
$PR750DEF          1           77     (2)      77     (2)
$PRDEF             4           84     (2)      84     (2)
$PSLDEF            2           83     (2)      83     (2)
$PUSHADR           1           365    (7)      365    (7)      366    (7)      367    (7)      368    (7)      369    (7)
                                               370    (7)      434    (8)      435    (8)      504    (10)     554    (12)
                                               556    (12)     617    (14)
$VIELD             1           67     (2)      67     (2)      82     (2)      83     (2)
$VIELD1            1           83     (2)      67     (2)      82     (2)      83     (2)
KA_DEF             2           86     (2)      86     (2)

                                               +-------------------------+
                                               ! Performance indicators !
                                               +-------------------------+

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                   129    00:00:00.30    00:00:00.73
Command processing               917    00:00:00.67    00:00:01.26
Pass 1                           704    00:00:09.23    00:00:16.56
Symbol table sort                  0    00:00:00.56    00:00:00.80
Pass 2                            17    00:00:01.86    00:00:02.87
Symbol table output                2    00:00:00.20    00:00:00.38
Psect synopsis output              2    00:00:00.03    00:00:00.03
Cross-reference output             6    00:00:00.80    00:00:01.15
Assembler run totals            1779    00:00:13.65    00:00:23.79

The working set limit was 2762 pages.
88846 bytes (174 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 603 non-local and 6 local symbols.
625 source lines were read in Pass 1, producing 34 object records in Pass 2.
125 pages of virtual memory were used to define 40 macros.

                                              +--------------------------+
                                              ! Macro library statistics !
                                              +--------------------------+

Macro library name                             Macros defined
------------------                             --------------
DISK$DIAGPACK02:[TAYLOR.ECKAX]ECKAX.MLB;1                 1
SYS$COMMON:[SYSLIB]DIAG.MLB;955                          38
SYS$COMMON:[SYSLIB]STARLET.MLB;1                         11
TOTALS (all libraries)                                   50

947 GETS were required to define 50 macros.

There were no errors, warnings or information messages.

MACRO/LIS=ECKAX1.LIS/CROSS ECKAX1
```

Table of contents

(1)   1   ECKAX2: RESERVED PROCESSOR REGISTERS
(2)   68   DECLARATIONS
(3)   118   Data Tables
(4)   193   ERROR MESSAGES
(5)   226   SUBROUTINES AND SERVICE ROUTINES
(6)   240   TEST 1: Reserved Processor Registers
(7)   258   SUBTEST 1.1: Reserved Processor Register Numbers
(8)   328   SUBTEST 1.2: Read only Processor Registers
(9)   394   SUBTEST 1.3: Write only Processor Registers
(10)   464   SUBTEST 1.4: Processor Register 28

```
0000        .SBTTL  ECKAX2: RESERVED PROCESSOR REGISTERS
0000        .TITLE  ECKAX2  VAX-11/750 Specific CPU Cluster Exerciser
0000        .IDENT  /5.2/
0000
0000  ;
0000  ; Copyright (C) 1980,1983,1986
0000  ; Digital Equipment Corporation, Maynard, Massachusetts 01754
0000  ;
0000  ; This software is furnished  under a license for use only on a single
0000  ; computer  system and  may be  copied only with  the inclusion of the
0000  ; above copyright notice.  This software, or any other copies thereof,
0000  ; may not be provided or  otherwise made available to any other person
0000  ; except for use on such system and to one who agrees to these license
0000  ; terms.  Title to and  ownership of the  software  shall at all times
0000  ; remain in DEC.
0000  ;
0000  ; The information in this software is subject to change without notice
0000  ; and should  not be construed  as a commitment  by Digital  Equipment
0000  ; Corporation.
0000  ;
0000  ; DEC assumes  no  responsibility  for  the use or  reliability of its
0000  ; software on equipment which is not supplied by DEC.
0000  ;
0000
0000  ;++
0000  ; Facility:     VAX Architecture Diagnostic.
0000  ;
0000  ; Abstract:     This module tests that all processor register numbers not
0000  ;               implemented on the VAX-11/750 will cause a reseved operand
0000  ;               fault for a MTPR or MFPR instruction. Also internal processor
0000  ;               registers specific to the VAX-11/750 which are write only and
0000  ;               read only are checked to not cause reserved operand faults
0000  ;               when writing and reading respectively and do in the converse
0000  ;               situation.
0000  ;
0000  ;
0000  ; Environment:  VAX Diagnostic Supervisor.
0000  ;
0000  ; Author:       Fred Roemer  19-Feb-81  Version 2.0
0000  ;
0000  ;       NO CHANGES TO THIS MODULE FOR VERSION 3.3 (DOCUMENT CHANGE ONLY)
0000  ;
0000  ;
0000  ;       Dan Milleville  19-Mar-82              Version 3.4
0000  ;       Changed Test 3 exit code to accomodate temporarily a hardware problem.
0000  ;
0000  ;       Dave Shull      30-Nov-82              Version 4.0
0000  ;       Added tests 13 and 14 for coverage of PCS module.
0000  ;
0000  ;       Tai-Chun Pan    04-Mar-83              Version 5.0
0000  ;       Added test 5 for coverage of Processor Halts Architecture Undefined.
0000  ;       Added tests 11 and 12 for coverage of new ROM with MRSP of TU58.
0000  ;
0000  ;       Dan Milleville  21-Nov-83              Version 5.1
0000  ;       Fixed error in timer routine (didn't turn off timer).
```

```
                    0000     56 ;
                    0000     57 ;       Wayne R. Taylor February 1986          Version 5.2
                    0000     58 ;       Redefine IPR's to jive with starlet.
                    0000     59 ;       Modified the code for problem report Eckax5 where:
                    0000     60 ;               Tests 6 & 13 tests with cache off.
                    0000     61 ;               Subtest 1 does not check bit 0 of the memory under test.
                    0000     62 ;               Subtest 4 tests row 0, 1, 2, but not row 3.
                    0000     63 ;       Modifications were supplied by the submitter.
                    0000     64 ;       Search on 5.2.
                    0000     65 ;
                    0000     66 ;--
                    0000     68          .SBTTL  DECLARATIONS
                    0000     69          .LIST   MEB
                    0000     70          .NLIST  CND
                00000000     71          .PSECT  ECKAXA, PAGE, NOWRT
                    0000     72          .LIBRARY         "SYS$LIBRARY:DIAG"; VAX family diagnostic library.
                    0000     73          $DS_BGNMOD       CEP_REPAIR, TN=1
                    0000        ;;;      Macro-32 Diagnostic macro library Version 8.3  DIAG.MLB (245)
                    0000     74
                    0000     75 ;
                    0000     76 ; Include files:
                    0000     77 ;
                    0000     78
                    0000     79          .LIBRARY         ECKAX'          ; CPU Cluster Exerciser library.
                    0000     80
                    0000     81
                    0000     82 ;
                    0000     83 ; Equated symbols:
                    0000     84 ;
                    0000     88
                    0000     89          $DS_PARDEF
                    0000     90          $DS_PSLDEF
                    0000     91          $PRDEF
                    0000     92          $PR750DEF       ; 5.2
                    0000     93          $DS_SCBDEF
                    0000     94          $DS_DSDEF
                    0000     95          $DS_DSADEF
                    0000     96          $DS_DSSDEF
                    0000     97          $DS_HPODEF
                    0000     98          KA_DEF
                    0000     99          $DS_ERRDEF
                    0000    100          ECKAX_SECDEF
                    0000    101
                    0000    102 ;
                    0000    103 ; Own storage:
                    0000    104 ;
                    0000    105
          00000004  0000    106 L_REG_ADD:      .BLKL           ; contains the register address
          00000005  0004    107 B_FLAGS:        .BLKB           ; contains the following flags
                    0005    108
          00000000  0005    109 V_NO_FAULT=0                    ; bit position - fault did not occur
          00000001  0005    110 M_NO_FAULT=1@0                  ; bit mask - expected fault did not occur
          00000001  0005    111 V_FAULT=1                       ; bit position - fault sucess flag
          00000002  0005    112 M_FAULT=1@1                     ; bit mask - fault sucess flag
                    0005    116
                0005   118          .SBTTL  Data Tables
                0005   119
                0005   120 ; This is a table that contains the unused internal processor register
                0005   121 ; numbers. This table is terminated by a -1.
                0005   122
      00000005  0005   123 ADDRESS_TABLE:  .LONG   ^X5
      00000006  0009   124                 .LONG   ^X6
      00000007  000D   125                 .LONG   ^X7
      0000000E  0011   126                 .LONG   ^XE
      0000000F  0015   127                 .LONG   ^XF
      00000016  0019   128                 .LONG   ^X16
      00000029  001D   129                 .LONG   ^X29
      0000002A  0021   130                 .LONG   ^X2A
      0000002B  0025   131                 .LONG   ^X2B
      0000002C  0029   132                 .LONG   ^X2C
      0000002D  002D   133                 .LONG   ^X2D
      0000002E  0031   134                 .LONG   ^X2E
      0000002F  0035   135                 .LONG   ^X2F
      00000030  0039   136                 .LONG   ^X30
      00000031  003D   137                 .LONG   ^X31
      00000032  0041   138                 .LONG   ^X32
      00000033  0045   139                 .LONG   ^X33
      00000034  0049   140                 .LONG   ^X34
      00000035  004D   141                 .LONG   ^X35
      00000036  0051   142                 .LONG   ^X36
      0000003F  0055   143                 .LONG   ^X3F
      0000003C  0059   144                 .LONG   ^X3C
      00000040  005D   145                 .LONG   ^X40
      00000080  0061   146                 .LONG   ^X80
      00000100  0065   147                 .LONG   ^X100
      00000200  0069   148                 .LONG   ^X200
      00000400  006D   149                 .LONG   ^X400
      00000800  0071   150                 .LONG   ^X800
      00001000  0075   151                 .LONG   ^X1000
      00002000  0079   152                 .LONG   ^X2000
      00004000  007D   153                 .LONG   ^X4000
      00008000  0081   154                 .LONG   ^X8000
      00010000  0085   155                 .LONG   ^X10000
      00020000  0089   156                 .LONG   ^X20000
      00040000  008D   157                 .LONG   ^X40000
      00080000  0091   158                 .LONG   ^X80000
      00100000  0095   159                 .LONG   ^X100000
      00200000  0099   160                 .LONG   ^X200000
      00400000  009D   161                 .LONG   ^X400000
      00800000  00A1   162                 .LONG   ^X800000
      01000000  00A5   163                 .LONG   ^X1000000
      02000000  00A9   164                 .LONG   ^X2000000
      04000000  00AD   165                 .LONG   ^X4000000
      08000000  00B1   166                 .LONG   ^X8000000
      10000000  00B5   167                 .LONG   ^X10000000
      20000000  00B9   168                 .LONG   ^X20000000
      40000000  00BD   169                 .LONG   ^X40000000
      80000000  00C1   170                 .LONG   ^X80000000
      FFFFFFFF  00C5   171                 .LONG   ^XFFFFFFFF
                00C9   172
```

```
                    00C9   173
                    00C9   174 ; This is a table that contains the write only internal processor register
                    00C9   175 ; numbers. This table is terminated by a -1.
                    00C9   176
                    00C9   177 WRITE_TABLE:
          00000037  00C9   178                 .LONG   ^X37
          0000003F  00CD   179                 .LONG   ^X3F
          FFFFFFFF  00D1   180                 .LONG   ^XFFFFFFFF
                    00D5   181
                    00D5   182
                    00D5   183 ; This is a table that contains the read only internal processor register
                    00D5   184 ; numbers. This table is terminated by a -1.
                    00D5   185
                    00D5   186 READ_TABLE:
          00000017  00D5   187                 .LONG   ^X17
          FFFFFFFF  00D9   188                 .LONG   ^XFFFFFFFF
                    00DD   189
                    00DD   190
                    00DD   191
```

```
                                         00DD     193             .SBTTL  ERROR MESSAGES
                                         00DD     194 ;------------------------------------------------------------------
                                         00DD     195 T_NO_FAULT_TO:
45 52 20 64 65 74 63 65 70 78 45 00'     00DD     196             .ASCIC    Expected RESERVED OPERAND FAULT did not occur durring!/ -
41 52 45 50 4F 20 44 45 56 52 45 53      00E9
64 69 64 20 54 4C 55 41 46 20 44 4E      00F5
64 20 72 75 63 63 6F 20 74 6F 6E 20      0101
65 76 6F 6D 2F 21 67 6E 69 72 72 75      010D
67 65 6C 6C 69 20 6E 61 20 6F 74 20      0119
72 6F 73 73 65 63 6F 72 70 20 6C 61      0125
64 61 20 72 65 74 73 69 67 65 72 20      0131
               2F 21 73 73 65 72 64      013D
                                  66     00DD
                                         0144     197                       move to an illegal processor register address!/
                                         0144     198 ;------------------------------------------------------------------
                                         0144     199 T_FAULT_ERR_F:
20 64 65 74 63 65 70 78 65 6E 55 00'     0144     200             .ASCIC    Unexpected RESERVED OPERAND FAULT occured while attempting!/ -
45 50 4F 20 44 45 56 52 45 53 45 52      0150
6F 20 54 4C 55 41 46 20 44 4E 41 52      015C
65 6C 69 68 77 20 64 65 72 75 63 63      0168
21 67 6E 69 74 70 6D 65 74 74 61 20      0174
                                  2F     0180
20 61 20 6E 6F 20 52 50 46 4D 20 61      0181     201                       a MFPR on a legal processor register address!/
73 65 63 6F 72 70 20 6C 61 67 65 6C      018D
72 65 74 73 69 67 65 72 20 72 6F 73      0199
      2F 21 73 73 65 72 64 64 61 20      01A5
                                  6A     0144
                                         01AF     202 ;------------------------------------------------------------------
                                         01AF     203 T_NO_FAULT_FROM:
45 52 20 64 65 74 63 65 70 78 45 00'     01AF     204             .ASCIC    Expected RESERVED OPERAND FAULT did not occur durring!/ -
41 52 45 50 4F 20 44 45 56 52 45 53      01BB
64 69 64 20 54 4C 55 41 46 20 44 4E      01C7
64 20 72 75 63 63 6F 20 74 6F 6E 20      01D3
            2F 21 67 6E 69 72 72 75      01DF
6E 61 20 6D 6F 72 66 20 65 76 6F 6D      01E7     205                       move from an illegal processor register address!/
6F 72 70 20 6C 61 67 65 6C 6C 69 20      01F3
73 69 67 65 72 20 72 6F 73 73 65 63      01FF
21 73 73 65 72 64 64 61 20 72 65 74      020B
                                  2F     0217
                                  68     01AF
                                         0218     206 ;------------------------------------------------------------------
                                         0218     207 T_FAULT_ERR_T:
20 64 65 74 63 65 70 78 65 6E 55 00'     0218     208             .ASCIC    Unexpected RESERVED OPERAND FAULT occured while attempting!/ -
45 50 4F 20 44 45 56 52 45 53 45 52      0224
6F 20 54 4C 55 41 46 20 44 4E 41 52      0230
65 6C 69 68 77 20 64 65 72 75 63 63      023C
21 67 6E 69 74 70 6D 65 74 74 61 20      0248
                                  2F     0254
20 61 20 6E 6F 20 52 50 54 4D 20 61      0255     209                       a MTPR on a legal processor register address!/
73 65 63 6F 72 70 20 6C 61 67 65 6C      0261
72 65 74 73 69 67 65 72 20 72 6F 73      026D
      2F 21 73 73 65 72 64 64 61 20      0279
                                  6A     0218
                                         0283     210 ;------------------------------------------------------------------
                                         0283     211 T_FPA_ERR:
```

```
6F 70 20 67 6E 69 74 61 6F 6C 46 00' 0283    212         .ASCIC   Floating point accelerator type does not match type in <13><10>-
61 72 65 6C 65 63 63 61 20 74 6E 69  028F
65 6F 64 20 65 70 79 74 20 72 6F 74  029B
20 68 63 74 61 6D 20 74 6F 6E 20 73  02A7
            0D 6E 69 20 65 70 79 74  02B3
72 6F 20 2C 65 6C 62 61 74 20 50 0A  02BB    213                  P table, or IPR 28 is not operating properly <13><10>
6E 20 73 69 20 38 32 20 52 50 49 20  02C7
67 6E 69 74 61 72 65 70 6F 20 74 6F  02D3
   0A 0D 79 6C 72 65 70 6F 72 70 20  02DF
                                 66  0283
                                     02EA    214 ;-----------------------------------------------------------------------------
                                     02EA    215 T_REG_ADD:
41 20 72 65 74 73 69 67 65 52 20 00' 02EA    216         .ASCIC    Register Addressed  =   ^X  !XL    !/
20 3D 20 20 64 65 73 73 65 72 64 64  02F6
20 20 20 4C 58 21 20 20 58 5E 20 20  0302
                              2F 21  030E
                                 25  02EA
                                     0310    217 ;-----------------------------------------------------------------------------
                                     0310    218
                                     0310    219 ERR_MESSAGE:
                                     0310    220         $DS_BGNMESSAGE
                               0000  0310                        .WORD     ^M<>                ; ENTRY MASK
                                     0312    221         $DS_PRINTB_S      FORMAT=@ERR$_P1(AP)
                      14 BC     9F   0312                        PUSHAB   @ERR$_P1(AP)
             000100E0 9F    01  FB   0315                        CALLS    #$$N, @#DS$PRINTB
                                     031C    222         $DS_PRINTB_S      FORMAT=T_REG_ADD,P0=L_REG_ADD
                    FCE0 CF     DD   031C                        PUSHL    L_REG_ADD
                      C7 AF     9F   0320                        PUSHAB   T_REG_ADD
             000100E0 9F    02  FB   0323                        CALLS    #$$N, @#DS$PRINTB
                                     032A    223         $DS_ENDMESSAGE
                                 04  032A                        RET                           ; RETURN TO DIAGNOSTIC SUPERVISOR
                                     032B    224
                                 032B   226           .SBTTL SUBROUTINES AND SERVICE ROUTINES
                                 032B   227
                                 032B   228
                                 032B   229 ; This routine will service all reserved operand faults.
                                 032B   230
                                 032B   231
                                 032B   232           .ALIGN  LONG
                                 032C   233 FAULT_SERVICE:
                6E      57    D0 032C   234           MOVL    R7,(SP)                    ; put return address on stack
           FCD0 CF      02    88 032F   235           BISB2   #M_FAULT,B_FLAGS           ; set the fault flag
                              02 0334   236           REI                                ; return
                                 0335   237
                                 0335   238           $DS_PAGE
                0335                .SBTTL  TEST 1:  Reserved Processor Registers
            00000000                        .PSECT  TEST_001, PAGE, NOWRT
                0020
                0020    241 ;++
                0020    242 ; Test description:
                0020    243 ;
                0020    244 ;       This test checks that processor numbers not used by the VAX-11/750
                0020    245 ;       cause reserve operand faults for the MFPR and MTPR instructions.
                0020    246 ;       Also processor numbers 17, 28 and 37 (X) are checked to not cause
                0020    247 ;       reserved operand faults when accessed.
                0020    248 ;
                0020    249 ; Assumptions:
                0020    250 ;
                0020    251 ;
                0020    252 ;--
                0020    253
                0020    254         $DS_BGNTEST     <IPR,ALL,DEFAULT>
                0020        DATA_001:
   00000000     0020                        .LONG   0               ; TEST ARGUMENT TABLE TERMINATOR
                0024        TEST_001::
       0000     0024                        .WORD   ^M<>            ; ENTRY MASK
                0026    255
                0026    256         $DS_PAGE
```

```
                                   0026              .SBTTL  SUBTEST 1.1:  Reserved Processor Register Numbers
                                   0026     259 ;+
                                   0026     260 ; Subtest description:
                                   0026     261 ;
                                   0026     262 ;        This subtest checks all the reserved processor register numbers that
                                   0026     263 ;        are not used in the VAX-11/750 processor.
                                   0026     264 ;
                                   0026     265 ; Subtest steps:
                                   0026     266 ;        SUBTEST RESERVED PROCESSOR REGISTER NUMBERS
                                   0026     267 ;        :  Initialize pointer to address table
                                   0026     268 ;        :  Get reserved operand fault vector to handle exception
                                   0026     269 ;        :  REPEAT
                                   0026     270 ;        :  :  Clear error flags
                                   0026     271 ;        :  :  Get address of register to attempt access
                                   0026     272 ;        :  :  Attempt MTPR on register
                                   0026     273 ;        :  :  Check for errors
                                   0026     274 ;        :  :  IF any errors
                                   0026     275 ;        :  :  :  THEN
                                   0026     276 ;        :  :  :  Report error
                                   0026     277 ;        :  :  :  Exit test
                                   0026     278 ;        :  :  ENDIF
                                   0026     279 ;        :  :  Attempt MFPR on register
                                   0026     280 ;        :  :  Check for errors
                                   0026     281 ;        :  :  IF any errors
                                   0026     282 ;        :  :  :  THEN
                                   0026     283 ;        :  :  :  Report error
                                   0026     284 ;        :  :  :  Exit test
                                   0026     285 ;        :  :  ENDIF
                                   0026     286 ;        :  : UNTIL All unused addresses done
                                   0026     287 ;        :  ENDREPEAT
                                   0026     288 ;        ENDSUBTEST RESERVED PROCESSOR REGISTER NUMBERS
                                   0026     289 ;
                                   0026     290 ;
                                   0026     291 ;
                                   0026     292 ;
                                   0026     293 ;-
                                   0026     294          $DS_BGNSUB
                                   0026         T1_S1::
     00010030 9F   00000000'EF   FA 0026                   CALLG   $S$, @#DS$BGNSUB
                                   0031     295          $DS_SETVEC_S    VECTOR=#SCB$L_ROPRAND   SRVADR=FAULT_SERVICE
                           00    DD 0031                   PUSHL   #0
                   0000032C'EF   DF 0033                   PUSHAL  FAULT_SERVICE
                           18    DD 0039                   PUSHL   #SCB$L_ROPRAND
          00010160 9F      03    FB 003B                   CALLS   #3, @#DS$SETVEC
          53    00000005'EF  9E 0042     296          MOVAB   ADDRESS_TABLE,R3         ; load R3 with address of table
          57    0000006A'EF  9E 0049     297 NEXT:    MOVAB   RENTRY_1,R7              ; load R7 with address of rentry point
                00000004'EF  94 0050     298          CLRB    B_FLAGS                  ; initialize error flag
                     55    63    D0 0056     299          MOVL    (R3),R5                  ; load address of IPR to access
          00000000'EF      55    D0 0059     300          MOVL    R5,L_REG_ADD             ; save register address for error
                     55    01    DA 0060     301          MTPR    #1,R5                    ; access IPR (should cause fault)
          00000004'EF      01    88 0063     302          BISB2   #M_NO_FAULT,B_FLAGS      ; flag no fault error
                                   006A     303 RENTRY_1:
       19 00000004'EF      00    E1 006A     304          BBC     #V_NO_FAULT,B_FLAGS,3$   ; br = MTPR did cause fault
                                   0072     305          $DS_ERRHARD_S   UNIT=#0,PRLINK=ERR_MESSAGE,P1=#T_NO_FAULT_TO
                   000000DD'8F   DD 0072                   PUSHL   #T_NO_FAULT_TO
```

```
                 00000310'EF    DF  0078                          PUSHAL  ERR_MESSAGE
                          00    DD  007E                          PUSHL   #0
                          00    DD  0080                          PUSHL   #0
                          01    DD  0082                          PUSHL   #$ER
                                    0084             ;;; TEST 1, SUBTEST 1, ERROR 1
            000100D0 9F   05    FB  0084                          CALLS   #$$M, @#DS$ERRHARD
            57   000000A5'EF    9E  008B   306 3$:       MOVAB    RENTRY_2,R7               ; load R7 with address of rentry point
                 00000004'EF    94  0092   307           CLRB     B_FLAGS                   ; initialize error flag
                       55 83    D0  0098   308           MOVL     (R3)+,R5                  ; load address of IPR to access
                       56 55    DB  009B   309           MFPR     R5,R6                     ; access IPR (should cause fault)
            00000004'EF   01    88  009E   310           BISB2    #M_NO_FAULT,B_FLAGS       ; set no fault error flag
                                    00A5   311 RENTRY_2:
         19 00000004'EF   00    E1  00A5   312           BBC      #V_NO_FAULT,B_FLAGS,5$   ; br = MFPR did cause fault
                                    00AD   313           $DS_ERRHARD_S   UNIT=#0,PRLINK=ERR_MESSAGE,P1=#T_NO_FAULT_FROM
                 000001AF'8F    DD  00AD                          PUSHL   #T_NO_FAULT_FROM
                 00000310'EF    DF  00B3                          PUSHAL  ERR_MESSAGE
                          00    DD  00B9                          PUSHL   #0
                          00    DD  00BB                          PUSHL   #0
                          02    DD  00BD                          PUSHL   #$ER
                                    00BF             ;;; TEST 1, SUBTEST 1, ERROR 2
            000100D0 9F   05    FB  00BF                          CALLS   #$$M, @#DS$ERRHARD
                       63 3F    D1  00C6   314 5$:       CMPL     #^X3F,(R3)                ; At register 3F?
                          0B    12  00C9   315           BNEQ     7$                        ; Branch if no
            0000FE15'EF   3E    91  00CB   316           CMPB     #62,DSA$GL_SID+1          ; Is this a rev 62 or higher machine?
                          02    14  00D2   317           BGTR     7$                        ; branch if no
                          83    D5  00D4   318           TSTL     (R3)+                     ; Skip register 3F
            63   FFFFFFFF 8F    D1  00D6   319 7$:       CMPL     #-1,(R3)                  ; check for end of table
                          03    13  00DD   320           BEQL     8$                        ; br = finished with all addresses
                        FF67    31  00DF   321           BRW      NEXT                      ; go do next address
                                    00E2   322 8$:       $DS_CLRVEC_S     VECTOR=#SCB$L_ROPRAND
                          18    DD  00E2                          PUSHL    #SCB$L_ROPRAND
            00010168 9F   01    FB  00E4                          CALLS    #1, @#DS$CLRVEC
                                    00EB   323
                                    00EB   324           $DS_ENDSUB
                                    00EB       T1_S1_X:
   00010038 9F   00000000'EF    FA  00EB                          CALLG    $S$, @#DS$ENDSUB
                                    00F6   325
                                    00F6   326           $DS_PAGE
```

```
                                      00F6               .SBTTL  SUBTEST 1.2:  Read only Processor Registers
                                      00F6   329 ;+
                                      00F6   330 ; Subtest description:
                                      00F6   331 ;
                                      00F6   332 ;       This subtest checks processor registers which are read only
                                      00F6   333 ;       registers. These are all processor specific registers.
                                      00F6   334 ;       This subtest does a read and checks that the read is
                                      00F6   335 ;       successful and a reserved operand fault does not occur. Then an
                                      00F6   336 ;       attempt to write the register is made which should cause a reserved
                                      00F6   337 ;       operand fault. If an errors are encountered they are reported.
                                      00F6   338 ;
                                      00F6   339 ;
                                      00F6   340 ; Subtest steps:
                                      00F6   341 ;
                                      00F6   342 ;       SUBTEST READ ONLY PROCESSOR REGISTERS
                                      00F6   343 ;       :  Get reserved operand fault vector to handle exception
                                      00F6   344 ;       :  REPEAT
                                      00F6   345 ;       :  :  Clear error flags
                                      00F6   346 ;       :  :  Attempt MTPR on register
                                      00F6   347 ;       :  :  Check for errors
                                      00F6   348 ;       :  :  IF any errors
                                      00F6   349 ;       :  :  :  THEN
                                      00F6   350 ;       :  :  :  Report error
                                      00F6   351 ;       :  :  :  Exit test
                                      00F6   352 ;       :  :  ENDIF
                                      00F6   353 ;       :  :  Attempt MFPR on register
                                      00F6   354 ;       :  :  Check for errors
                                      00F6   355 ;       :  :  IF any errors
                                      00F6   356 ;       :  :  :  THEN
                                      00F6   357 ;       :  :  :  Report error
                                      00F6   358 ;       :  :  :  Exit test
                                      00F6   359 ;       :  :  ENDIF
                                      00F6   360 ;       :  : UNTIL All read only registers are done
                                      00F6   361 ;       :  ENDREPEAT
                                      00F6   362 ;       ENDSUBTEST READ ONLY PROCESSOR REGISTERS
                                      00F6   363 ;
                                      00F6   364 ;
                                      00F6   365 ;
                                      00F6   366 ;-
                                      00F6   367         $DS_BGNSUB
                                      00F6       T1_S2::
     00010030 9F    0000000C'EF   FA  00F6                       CALLG   $S$, @#DS$BGNSUB
              53    000000D5'EF   9E  0101   368         MOVAB   READ_TABLE,R3            ; load R3 with address of table
                                      0108   369         $DS_SETVEC_S    VECTOR=#SCB$L_ROPRAND   SRVADR=FAULT_SERVICE
                             00   DD  0108                       PUSHL   #0
                    0000032C'EF   DF  010A                       PUSHAL  FAULT_SERVICE
                             18   DD  0110                       PUSHL   #SCB$L_ROPRAND
              00010160 9F    03   FB  0112                       CALLS   #3, @#DS$SETVEC
                       55    63   D0  0119   370 NEXT1:  MOVL    (R3),R5                  ; load address of IPR to access
              00000000'EF    55   D0  011C   371         MOVL    R5,L_REG_ADD             ; save register address for error
              57    0000013A'EF   9E  0123   372         MOVAB   RENTRY_3,R7              ; load R7 with address of rentry point
                    00000004'EF   94  012A   373         CLRB    B_FLAGS                  ; initialize error flag
                       55    01   DA  0130   374         MTPR    #1,R5                    ; access IPR (should cause fault)
              00000004'EF    01   88  0133   375         BISB2   #M_NO_FAULT,B_FLAGS      ; set no fault error flag
                                      013A   376 RENTRY_3:
```

```
           19 00000004'EF   00   E1  013A   377         BBC     #V_NO_FAULT,B_FLAGS,3$  ; br = MTPR did cause fault
                                     0142   378         $DS_ERRHARD_S   UNIT=#0,PRLINK=ERR_MESSAGE,P1=#T_NO_FAULT_TO
                   000000DD'8F   DD  0142                       PUSHL   #T_NO_FAULT_TO
                   00000310'EF   DF  0148                       PUSHAL  ERR_MESSAGE
                            00   DD  014E                       PUSHL   #0
                            00   DD  0150                       PUSHL   #0
                            01   DD  0152                       PUSHL   #$ER
                                     0154       ;;; TEST 1, SUBTEST 2, ERROR 1
              000100D0 9F   05   FB  0154                       CALLS   #$$M, @#DS$ERRHARD
              57   0000016B'EF   9E  015B   379 3$:     MOVAB   RENTRY_4,R7             ; load R7 with address of rentry point
                   00000004'EF   94  0162   380         CLRB    B_FLAGS                 ; initialize error flag
                       56   55   DB  0168   381         MFPR    R5,R6                   ; access IPR (should not cause fault)
                                     016B   382 RENTRY_4:
           19 00000004'EF   01   E1  016B   383         BBC     #V_FAULT,B_FLAGS,5$     ; br = exception rountine executed
                                     0173   384         $DS_ERRHARD_S   UNIT=#0,PRLINK=ERR_MESSAGE,P1=#T_FAULT_ERR_F
                   00000144'8F   DD  0173                       PUSHL   #T_FAULT_ERR_F
                   00000310'EF   DF  0179                       PUSHAL  ERR_MESSAGE
                            00   DD  017F                       PUSHL   #0
                            00   DD  0181                       PUSHL   #0
                            02   DD  0183                       PUSHL   #$ER
                                     0185       ;;; TEST 1, SUBTEST 2, ERROR 2
              000100D0 9F   05   FB  0185                       CALLS   #$$M, @#DS$ERRHARD
                            83   D5  018C   385 5$:     TSTL    (R3)+                   ; Point at next register address
              63   FFFFFFFF 8F   D1  018E   386         CMPL    #-1,(R3)                ; check for end of table
                            03   13  0195   387         BEQL    8$                      ; br = finished with all addresses
                          FF7F   31  0197   388         BRW     NEXT1                   ; go do next address
                                     019A   389 8$:     $DS_CLRVEC_S    VECTOR=#SCB$L_ROPRAND
                            18   DD  019A                       PUSHL   #SCB$L_ROPRAND
              00010168 9F   01   FB  019C                       CALLS   #1, @#DS$CLRVEC
                                     01A3   390         $DS_ENDSUB
                                     01A3       T1_S2_X:
     00010038 9F   0000000C'EF   FA  01A3                       CALLG   $S$, @#DS$ENDSUB
                                     01AE   391
                                     01AE   392         $DS_PAGE
```

```
                                              01AE              .SBTTL  SUBTEST 1.3:  Write only Processor Registers
                                              01AE     395 ;+
                                              01AE     396 ; Subtest description:
                                              01AE     397 ;
                                              01AE     398 ;        This subtest checks processor registers which are write only
                                              01AE     399 ;        registers. These are processor specific registers. This subtest does
                                              01AE     400 ;        a write and checks that it is successful and a reserved operand fault
                                              01AE     401 ;        does not occur. Then an attempt to read the register is made which
                                              01AE     402 ;        should cause a reserved operand fault. If an errors are encountered
                                              01AE     403 ;        they are reported.
                                              01AE     404 ;
                                              01AE     405 ;
                                              01AE     406 ; Subtest steps:
                                              01AE     407 ;        SUBTEST WRITE ONLY PROCESSOR REGISTERS
                                              01AE     408 ;        :  Get reserved operand fault vector to handle exception
                                              01AE     409 ;        :  Clear error flags
                                              01AE     410 ;        :  REPEAT
                                              01AE     411 ;        :  :  Attempt MTPR on register
                                              01AE     412 ;        :  :  Check for errors
                                              01AE     413 ;        :  :  IF any errors
                                              01AE     414 ;        :  :  :  THEN
                                              01AE     415 ;        :  :  :  Report error
                                              01AE     416 ;        :  :  :  Exit test
                                              01AE     417 ;        :  :  ENDIF
                                              01AE     418 ;        :  :  Attempt MFPR on register
                                              01AE     419 ;        :  :  Check for errors
                                              01AE     420 ;        :  :  IF any errors
                                              01AE     421 ;        :  :  :  THEN
                                              01AE     422 ;        :  :  :  Report error
                                              01AE     423 ;        :  :  :  Exit test
                                              01AE     424 ;        :  :  ENDIF
                                              01AE     425 ;        :  : UNTIL all write only registers are done
                                              01AE     426 ;        :  ENDREPEAT
                                              01AE     427 ;        ENDSUBTEST WRITE ONLY PROCESSOR REGISTERS
                                              01AE     428 ;
                                              01AE     429 ;
                                              01AE     430 ;
                                              01AE     431 ;-
                                              01AE     432              $DS_BGNSUB
                                              01AE          T1_S3::
        00010030 9F    00000018'EF    FA      01AE                          CALLG   $S$, @#DS$BGNSUB
                                              01B9     433              $DS_SETVEC_S    VECTOR=#SCB$L_ROPRAND   SRVADR=FAULT_SERVICE
                                00    DD      01B9                          PUSHL   #0
                      0000032C'EF    DF      01BB                          PUSHAL  FAULT_SERVICE
                                18    DD      01C1                          PUSHL   #SCB$L_ROPRAND
                  00010160 9F   03    FB      01C3                          CALLS   #3, @#DS$SETVEC
                  53   000000C9'EF    9E      01CA     434      MOVAB   WRITE_TABLE,R3           ; load R3 with address of table
                                              01D1     435              $DS_SETVEC_S    VECTOR=#SCB$L_ROPRAND   SRVADR=FAULT_SERVICE
                                00    DD      01D1                          PUSHL   #0
                      0000032C'EF    DF      01D3                          PUSHAL  FAULT_SERVICE
                                18    DD      01D9                          PUSHL   #SCB$L_ROPRAND
                  00010160 9F   03    FB      01DB                          CALLS   #3, @#DS$SETVEC
                           55   83    D0      01E2     436 NEXT2:   MOVL    (R3)+,R5                 ; load address of IPR to access
                  00000000'EF   55    D0      01E5     437          MOVL    R5,L_REG_ADD             ; save register address for error
                  57   00000203'EF    9E      01EC     438          MOVAB   RENTRY_5,R7              ; load R7 with address of rentry point
```

```
           00000004'EF    94  01F3     439          CLRB     B_FLAGS                       ; initialize error flag
                56   55   DB  01F9     440          MFPR     R5,R6                         ; access IPR (should cause fault)
       00000004'EF   01   88  01FC     441          BISB2    #M_NO_FAULT,B_FLAGS           ; set no fault error flag
                              0203     442 RENTRY_5:
    19 00000004'EF   00   E1  0203     443          BBC      #V_NO_FAULT,B_FLAGS,3$  ; br = MFPR did cause fault
                              020B     444          $DS_ERRHARD_S    UNIT=#0,PRLINK=ERR_MESSAGE,P1=#T_NO_FAULT_TO
           000000DD'8F    DD  020B                           PUSHL    #T_NO_FAULT_TO
           00000310'EF    DF  0211                           PUSHAL   ERR_MESSAGE
                     00   DD  0217                           PUSHL    #0
                     00   DD  0219                           PUSHL    #0
                     01   DD  021B                           PUSHL    #$ER
                              021D          ;;; TEST 1, SUBTEST 3, ERROR 1
       000100D0 9F   05   FB  021D                           CALLS    #$$M, a#DS$ERRHARD
       57    00000234'EF  9E  0224     445 3$:      MOVAB    RENTRY_6,R7                   ; load R7 with address of rentry point
           00000004'EF    94  022B     446          CLRB     B_FLAGS                       ; initialize error flag
                55   01   DA  0231     447          MTPR     #1,R5                         ; access IPR (should not cause fault)
                              0234     448 RENTRY_6:
    19 00000004'EF   01   E1  0234     449          BBC      #V_FAULT,B_FLAGS,5$       ; br = no exception occured
                              023C     450          $DS_ERRHARD_S    UNIT=#0,PRLINK=ERR_MESSAGE,P1=#T_FAULT_ERR_T
           00000218'8F    DD  023C                           PUSHL    #T_FAULT_ERR_T
           00000310'EF    DF  0242                           PUSHAL   ERR_MESSAGE
                     00   DD  0248                           PUSHL    #0
                     00   DD  024A                           PUSHL    #0
                     02   DD  024C                           PUSHL    #$ER
                              024E          ;;; TEST 1, SUBTEST 3, ERROR 2
       000100D0 9F   05   FB  024E                           CALLS    #$$M, a#DS$ERRHARD
                63   3F   D1  0255     451 5$:      CMPL     #^X3F,(R3)                    ; At register 3F?
                     0B   12  0258     452          BNEQ     7$                            ; Branch if no
       0000FE15'EF   3E   91  025A     453          CMPB     #62,DSA$GL_SID+1              ; Is this lower than rev 62 machine?
                     02   19  0261     454          BLSS     7$                            ; branch if yes
                     83   D5  0263     455          TSTL     (R3)+                         ; Skip register 3F
       63    FFFFFFFF 8F  D1  0265     456 7$:      CMPL     #-1,(R3)                      ; check for end of table
                     03   13  026C     457          BEQL     8$                            ; br = finished with all addresses
                   FF71   31  026E     458          BRW      NEXT2                         ; go do next address
                              0271     459 8$:      $DS_CLRVEC_S     VECTOR=#SCB$L_ROPRAND
                     18   DD  0271                           PUSHL    #SCB$L_ROPRAND
       00010168 9F   01   FB  0273                           CALLS    #1, a#DS$CLRVEC
                              027A     460          $DS_ENDSUB
                              027A          T1_S3_X:
00010038 9F    00000018'EF    FA  027A                           CALLG    $S$, a#DS$ENDSUB
                              0285     461
                              0285     462          $DS_PAGE
```

```
                                     0285               .SBTTL  SUBTEST 1.4:  Processor Register 28
                                     0285   465 ;+
                                     0285   466 ; Subtest description:
                                     0285   467 ;
                                     0285   468 ;       This subtest checks processor register 28 which contains the floating
                                     0285   469 ;       point accelerator control and status information. An attempt to turn
                                     0285   470 ;       the accelerator on is made and then the status is compared to the
                                     0285   471 ;       entry in the P TABLE, setup by the operator. The type is checked and
                                     0285   472 ;       if any errors are detected they are reported.
                                     0285   473 ;
                                     0285   474 ;
                                     0285   475 ; Subtest steps:
                                     0285   476 ;       SUBTEST PROCESSOR REGISTER 28
                                     0285   477 ;       :  Get reserved operand fault vector to handle exception
                                     0285   478 ;       :  Clear error flags
                                     0285   479 ;       :  Attempt MTPR on register 28 to turn FPA on
                                     0285   480 ;       :  Check for reserved operand fault error
                                     0285   481 ;       :  IF an error
                                     0285   482 ;       :  :  THEN
                                     0285   483 ;       :  :  Report error
                                     0285   484 ;       :  :  Exit test
                                     0285   485 ;       :  ENDIF
                                     0285   486 ;       :  Attempt MFPR on register 28 to read type
                                     0285   487 ;       :  Check type read matches type in P TABLE
                                     0285   488 ;       :  IF an error
                                     0285   489 ;       :  :  THEN
                                     0285   490 ;       :  :  Report error
                                     0285   491 ;       :  :  Exit test
                                     0285   492 ;       :  ENDIF
                                     0285   493 ;       ENDSUBTEST PROCESSOR REGISTER 28
                                     0285   494 ;
                                     0285   495 ;
                                     0285   496 ;-
                                     0285   497         $DS_BGNSUB
                                     0285       T1_S4::
     00010030 9F   00000024'EF   FA  0285                   CALLG   $S$, @#DS$BGNSUB
                                     0290   498         $DS_SETVEC_S    VECTOR=#SCB$L_ROPRAND   SRVADR=FAULT_SERVICE
                             00  DD  0290                   PUSHL   #0
                   0000032C'EF   DF  0292                   PUSHAL  FAULT_SERVICE
                             18  DD  0298                   PUSHL   #SCB$L_ROPRAND
              00010160 9F    03  FB  029A                   CALLS   #3, @#DS$SETVEC
              57   000002BC'EF   9E  02A1   499         MOVAB   RENTRY_7,R7                 ; load R7 with address of rentry point
                   00000004'EF   94  02A8   500         CLRB    B_FLAGS                     ; initialize error flag
              00000000'EF    28  D0  02AE   501         MOVL    #^X28,L_REG_ADD             ; load reg address for error
              28   00008000 8F   DA  02B5   502         MTPR    #1@15,#PR750$_ACCS          ; turn on FPA (should not fault) 5.2
                                     02BC   503 RENTRY_7:
           19 00000004'EF    01  E1  02BC   504         BBC     #V_FAULT,B_FLAGS,3$         ; br = MTPR did not cause fault
                                     02C4   505         $DS_ERRHARD_S   UNIT=#0,PRLINK=ERR_MESSAGE,P1=#T_FAULT_ERR_T
                   00000218'8F   DD  02C4                   PUSHL   #T_FAULT_ERR_T
                   00000310'EF   DF  02CA                   PUSHAL  ERR_MESSAGE
                             00  DD  02D0                   PUSHL   #0
                             00  DD  02D2                   PUSHL   #0
                             01  DD  02D4                   PUSHL   #$ER
                                     02D6       ;;; TEST 1, SUBTEST 4, ERROR 1
              000100D0 9F    05  FB  02D6                   CALLS   #$$M, @#DS$ERRHARD
              57    000002ED'EF    9E  02DD    506 3$:      MOVAB   RENTRY_8,R7                ; load R7 with address of rentry point
                    00000004'EF    94  02E4    507          CLRB    B_FLAGS                    ; initialize error flag
                          58   28  DB  02EA    508          MFPR    #PR750$_ACCS,R8            ; read FPA type (should not fault) 5.2
                                       02ED    509 RENTRY_8:
           19 00000004'EF      01  E1  02ED    510          BBC     #V_FAULT,B_FLAGS,5$        ; br = MFPR did not cause fault
                                       02F5    511          $DS_ERRHARD_S   UNIT=#0,PRLINK=ERR_MESSAGE,P1=#T_FAULT_ERR_F
                    00000144'8F    DD  02F5                         PUSHL   #T_FAULT_ERR_F
                    00000310'EF    DF  02FB                         PUSHAL  ERR_MESSAGE
                               00  DD  0301                         PUSHL   #0
                               00  DD  0303                         PUSHL   #0
                               02  DD  0305                         PUSHL   #$ER
                                       0307         ;;; TEST 1, SUBTEST 4, ERROR 2
              000100D0 9F      05  FB  0307                         CALLS   #$$M, @#DS$ERRHARD
              59    00000000'EF    D0  030E    512 5$:      MOVL    KA_PTABLE,R9               ; load address of P TABLE
                     42 A9     58  91  0315    513          CMPB    R8,KA$B_ACC_TYPE(R9)       ; check if type is the same
                               13  13  0319    514          BEQL    7$                         ; br = yes, same type
                                       031B    515          $DS_ERRHARD_S   UNIT=#0,MSGADR=T_FPA_ERR
                               00  DD  031B                         PUSHL   #0
                    00000283'EF    DF  031D                         PUSHAL  T_FPA_ERR
                               00  DD  0323                         PUSHL   #0
                               03  DD  0325                         PUSHL   #$ER
                                       0327         ;;; TEST 1, SUBTEST 4, ERROR 3
              000100D0 9F      04  FB  0327                         CALLS   #$$M, @#DS$ERRHARD
                                       032E    516 7$:      $DS_CLRVEC_S    VECTOR=#SCB$L_ROPRAND
                               18  DD  032E                         PUSHL   #SCB$L_ROPRAND
              00010168 9F      01  FB  0330                         CALLS   #1, @#DS$CLRVEC
                                       0337    517
                                       0337    518          $DS_ENDSUB
                                       0337         T1_S4_X:
      00010038 9F   00000024'EF    FA  0337                         CALLG   $S$, @#DS$ENDSUB
                                       0342    519          $DS_ENDTEST
                          50   01  D0  0342                         MOVL    #1, R0          ; NORMAL EXIT
                                       0345         TEST_001_X::
              00010058 9F      6E  FA  0345                         CALLG   (SP), @#DS$BREAK
                                   04  034C                         RET                     ; RETURN TO TEST SEQUENCER
                                       034D    520          $DS_ENDMOD
                                   00000000                         .PSECT  $TSTCNT, NOEXE, NOWRT, OVR, LONG
                             00000001  0000                         .LONG   $TN
                                       0004    521          .END
```

```
$$$                             = 00000024 R      05        DS$K_SEVERE                        = 00000004
$$ARGS                          = 0000000A                  DS$K_SUBSYS                        = 00000066
$$E                             = 00000001                  DS$K_WARNING                       = 00000000
$$M                             = 00000004                  DS$LOAD                              00010198
$$N                             = 00000000                  DS$LOADPCS                           000101C0
$$S                             = FFFFFFFF                  DS$MAPDBGBLOCK                       00010118
$$T1                            = 0000002C                  DS$MEMSIZE                           000101D8
$ENV                            = 00000001  G               DS$MMOFF                             00010158
$ER                             = FFFFFFFF                  DS$MMON                              00010150
$MO                             = 00000001  G               DS$MOVPHY                            00010148
$S$                             = 00000024 R      05        DS$MOVVRT                            00010140
$ST                             = 00000000                  DS$PARSE                             000100B8
$TN                             = 00000001                  DS$PRINTB                            000100E0
AΓ)RESS_TABLE                     00000005 R      01        DS$PRINTF                            000100F0
ALL                             = 00000009  G               DS$PRINTS                            000100F8
BASE_001                          00000000 R      03        DS$PRINTSIG                          00010100
B_FLAGS                           00000004 R      01        DS$PRINTX                            000100E8
CEP_FUNCTIONAL                  = 00000000                  DS$PROBE                             000101A0
CEP_REPAIR                      = 00000001                  DS$RELBUF                            00010128
DATA_001                          00000020 R      03        DS$RELMEM                            00010138
DEFAULT                         = 00000000  G               DS$SERVICE_DISPATCHER                000101D0
DS$ABORT                          00010020                  DS$SETIPL                            00010178
DS$ASKADR                         00010090                  DS$SETMAP                            00010188
DS$ASKDATA                        00010080                  DS$SETPRIEXV                         00010110
DS$ASKLGCL                        00010098                  DS$SETVEC                            00010160
DS$ASKSTR                         000100A0                  DS$SHOCHAN                           00010190
DS$ASKVLD                         00010088                  DS$SUMMARY                           00010028
DS$ATTACH                         000101A8                  DS$WAITMS                            00010060
DS$BGNSUB                         00010030                  DS$WAITUS                            00010068
DS$BRANCH                         000100A8                  DS$_ARITH                          = 006600D0
DS$BREAK                          00010058                  DS$_ASBE                           = 00660118
DS$CANWAIT                        00010070                  DS$_BADLINK                        = 006600F0
DS$CHANNEL                        00010180                  DS$_BADTYPE                        = 006600E8
DS$CKLOOP                         00010040                  DS$_BIIC                           = 00660120
DS$CLRVEC                         00010168                  DS$_CHME                           = 006600A8
DS$CNTRLC                         00010078                  DS$_CHMK                           = 006600E0
DS$CVTREG                         000100B0                  DS$_DEVNAME                        = 00660108
DS$ENDPASS                        00010010                  DS$_ERROR                          = 00660002
DS$ENDSUB                         00010038                  DS$_FHWE                           = 00660068
DS$ERRDEV                         000100C8                  DS$_FRAGBUF                        = 00660080
DS$ERRHARD                        000100D0                  DS$_ICBUSY                         = 006600C8
DS$ERRPREP                        00010108                  DS$_ICERR                          = 006600C0
DS$ERRSOFT                        000100D8                  DS$_IHWE                           = 00660060
DS$ERRSYS                         000100C0                  DS$_ILLCHAR                        = 00660018
DS$ESCAPE                         00010050                  DS$_ILLPAGCNT                      = 00660078
DS$FREEDBGSYM                     000101B8                  DS$_ILLUNIT                        = 00660100
DS$GETBUF                         00010120                  DS$_INSFMEM                        = 00660050
DS$GETMEM                         00010130                  DS$_IPL2HI                         = 006600B8
DS$GETTERM                        000101C8                  DS$_IVADDR                         = 00660040
DS$GPHARD                         00010018                  DS$_IVVECT                         = 00660038
DS$HELP                           000101B0                  DS$_KRNLSTK                        = 00660090
DS$INITSCB                        00010170                  DS$_LOGIC                          = 00660070
DS$INLOOP                         00010048                  DS$_MCHK                           = 00660088
DS$K_ERROR                      = 00000002                  DS$_MMOFF                          = 00660058
DS$K_NORMAL                     = 00000001                  DS$_NEEDUNIT                       = 006600F8
```

```
DS$_NODE                        = 00660128          ERR$_P6                         = 00000028
DS$_NOPCS                       = 00660110          ERR$_POINTER                    = 00000010
DS$_NORMAL                      = 00660001          ERR$_UNIT                       = 00000008
DS$_NOSUPPORT                   = 006600B1          ERR_MESSAGE                       00000310 R       01
DS$_NOTDON                      = 00660030          FAULT_SERVICE                     0000032C R       01
DS$_NOTIMP                      = 006600B0          HALT                            = 00000004  G
DS$_NULLSTR                     = 00660010          HP$A_DEPENDENT                    00000032
DS$_OVERFLOW                    = 00660008          HP$A_DEVICE                       00000018
DS$_POWER                       = 00660098          HP$A_DVA                          0000001C
DS$_PROGERR                     = 00660020          HP$A_LINK                         00000020
DS$_SEVERE                      = 00660004          HP$B_DRIVE                        0000000B
DS$_TRANSL                      = 006600A0          HP$B_FLAGS                        0000000A
DS$_TRUNCATE                    = 00660028          HP$Q_DEVICE                       00000000
DS$_UNEXPINT                    = 006600D8          HP$T_DEVICE                       0000000C
DS$_VASFULL                     = 00660048          HP$T_TYPE                         00000026
DS$_WARNING                     = 00660000          HP$W_SIZE                         00000008
DSA$AL_APTMAIL                    0000FE00          HP$W_VECTOR                       00000024
DSA$AT_APTTXT                     0000FA00          IPR                             = 00000006  G
DSA$GL_APTCOM                     0000FE04          KA$B_ACC_TYPE                     00000042
DSA$GL_DEVLEN                     0000FE58          KA$B_SID_TYPE                     0000003D
DSA$GL_ERRNO                      0000FE44          KA$K_LEN                          00000043
DSA$GL_EVENT                      0000FE48          KA$L_FLAGS                        00000032
DSA$GL_FLAGS                      0000FE00          KA$L_SID                          0000003A
DSA$GL_MSGTYP                     0000FE40          KA$W_LATENCY                      0000003E
DSA$GL_PASSES                     0000FE08          KA$W_PROGRESS                     00000040
DSA$GL_PASSNO                     0000FE54          KA$W_RESERVED                     00000038
DSA$GL_SECTNO                     0000FE10          KA$W_WCS                          00000036
DSA$GL_SID                        0000FE14          KA_PTABLE                         ********   X   03
DSA$GL_SUBTNO                     0000FE4C          L_REG_ADD                         00000000 R       01
DSA$GL_TESTNO                     0000FE50          MANUAL                          = 00000001  G
DSA$GL_UNITS                      0000FE0C          MCHK                            = 00000005  G
DSA$GQ_MSGPTR                     0000FE68          MSG_001                           00000002 R       03
DSA$GT_DEVNAM                     0000FE5C          M_FAULT                         = 00000002
ENV$M_DOMAIN                    = 00000002          M_NO_FAULT                      = 00000001
ENV$M_LEVEL                     = 00000001          NEXT                              00000049 R       03
ENV$M_SUPER                     = 000003FC          NEXT1                             00000119 R       03
ENV$S_DOMAIN                    = 00000001          NEXT2                             000001E2 R       03
ENV$S_LEVEL                     = 00000001          PAR$M_ATDEF                     = 00000010
ENV$S_SUPER                     = 00000008          PAR$M_ATHI                      = 00000008
ENV$V_DOMAIN                    = 00000001          PAR$M_ATLO                      = 00000004
ENV$V_LEVEL                     = 00000000          PAR$M_NODEF                     = 00000002
ENV$V_SUPER                     = 00000002          PAR$V_ATDEF                     = 00000004
ENV$_CPU                        = 00000000          PAR$V_ATHI                      = 00000003
ENV$_FUNCTIONAL                 = 00000000          PAR$V_ATLO                      = 00000002
ENV$_REPAIR                     = 00000001          PAR$V_NODEF                     = 00000001
ENV$_SUPER                      = 00000001          PAR$_BIN                        = 00000002
ENV$_SYSTEM                     = 00000001          PAR$_DEC                        = 0000000A
ERR$_MSGADR                     = 0000000C          PAR$_HEX                        = 00000010
ERR$_NARGS                      = 0000000A          PAR$_NO                         = 00000000
ERR$_NUM                        = 00000004          PAR$_OCT                        = 00000008
ERR$_P1                         = 00000014          PAR$_YES                        = 00000001
ERR$_P2                         = 00000018          PCS                             = 00000008  G
ERR$_P3                         = 0000001C          POWER_FAIL                      = 00000003  G
ERR$_P4                         = 00000020          PR750$_ACCS                     = 00000028
ERR$_P5                         = 00000024          PSL$K_EXEC                      = 00000001
```

```
PSL$K_KERNEL                  = 00000000                SCB$L_ZERO                       00000000
PSL$K_SUPER                   = 00000002                SEP_FUNCTIONAL                 = 00000002
PSL$K_USER                    = 00000003                SEP_REPAIR                     = 00000003
PSL$M_CBIT                    = 00000001                SIZ...                         = 00000001
PSL$M_NBIT                    = 00000008                SYS$ALLOC                        00010238
PSL$M_VBIT                    = 00000002                SYS$ASCTIM                       00010248
PSL$M_ZBIT                    = 00000004                SYS$ASSIGN                       00010250
PSL$V_CBIT                    = 00000000                SYS$BINTIM                       00010258
PSL$V_NBIT                    = 00000003                SYS$CANCEL                       00010260
PSL$V_VBIT                    = 00000001                SYS$CANTIM                       00010268
PSL$V_ZBIT                    = 00000002                SYS$CLOSE                        000105B8
READ_TABLE                      000000D5 R    01        SYS$CLREF                        00010298
RENTRY_1                        0000006A R    03        SYS$CONNECT                      000105C0
RENTRY_2                        000000A5 R    03        SYS$DALLOC                       000102D8
RENTRY_3                        0000013A R    03        SYS$DASSGN                       000102E0
RENTRY_4                        0000016B R    03        SYS$DISCONNECT                   000105D0
RENTRY_5                        00000203 R    03        SYS$FAO                          00010350
RENTRY_6                        00000234 R    03        SYS$FAOL                         00010358
RENTRY_7                        000002BC R    03        SYS$GET                          00010580
RENTRY_8                        000002ED R    03        SYS$GETCHN                       000104C8
SCB$L_ACCESS                    00000020                SYS$GETTIM                       00010378
SCB$L_ARITH                     00000034                SYS$LKWSET                       000103A0
SCB$L_BREAK                     0000002C                SYS$NUMTIM                       000103B8
SCB$L_CHME                      00000044                SYS$OPEN                         00010608
SCB$L_CHMK                      00000040                SYS$QIO                          000103C8
SCB$L_CHMS                      00000048                SYS$QIOW                         00010200
SCB$L_CHMU                      0000004C                SYS$READ                         00010590
SCB$L_COMPAT                    00000030                SYS$READEF                       000103D0
SCB$L_KNLSTK                    00000008                SYS$SETAST                       000103F8
SCB$L_MACHCHK                   00000004                SYS$SETEF                        00010400
SCB$L_OPCCUS                    00000014                SYS$SETIMR                       00010420
SCB$L_OPCDEC                    00000010                SYS$SETPRI                       00010428
SCB$L_POWER                     0000000C                SYS$SETPRT                       00010430
SCB$L_RADRMOD                   0000001C                SYS$SETRWM                       00010438
SCB$L_ROPRAND                   00000018                SYS$SETSWM                       00010448
SCB$L_RXDB                      000000F8                SYS$ULKPAG                       00010460
SCB$L_SFTLVL1                   00000084                SYS$ULWSET                       00010468
SCB$L_SFTLVL10                  000000A8                SYS$UNWIND                       00010470
SCB$L_SFTLVL11                  000000AC                SYS$WAITFR                       00010478
SCB$L_SFTLVL12                  000000B0                SYS$WFLAND                       00010488
SCB$L_SFTLVL13                  000000B4                SYS$WFLOR                        00010490
SCB$L_SFTLVL14                  000000B8                T1_S1                            00000026 RG    03
SCB$L_SFTLVL15                  000000BC                T1_S1_X                          000000EB R     03
SCB$L_SFTLVL2                   00000088                T1_S2                            000000F6 RG    03
SCB$L_SFTLVL3                   0000008C                T1_S2_X                          000001A3 R     03
SCB$L_SFTLVL4                   00000090                T1_S3                            000001AE RG    03
SCB$L_SFTLVL5                   00000094                T1_S3_X                          0000027A R     03
SCB$L_SFTLVL6                   00000098                T1_S4                            00000285 RG    03
SCB$L_SFTLVL7                   0000009C                T1_S4_X                          00000337 R     03
SCB$L_SFTLVL8                   000000A0                TEST_001                         00000024 RG    03
SCB$L_SFTLVL9                   000000A4                TEST_001_X                       00000345 RG    03
SCB$L_TBIT                      00000028                TU58                           = 00000002  G
SCB$L_TIMER                     000000C0                T_FAULT_ERR_F                    00000144 R     01
SCB$L_TRANSL                    00000024                T_FAULT_ERR_T                    00000218 R     01
SCB$L_TXDB                      000000FC                T_FPA_ERR                        00000283 R     01
```

```
T_NO_FAULT_FROM                   000001AF R      01
T_NO_FAULT_TO                     000000DD R      01
T_REG_ADD                         000002EA R      01
V_FAULT                         = 00000001
V_NO_FAULT                      = 00000000
WCS                             = 00000007  G
WRITE_TABLE                       000000C9 R      01

                                                 +-----------------+
                                                 ! Psect synopsis !
                                                 +-----------------+

PSECT name                       Allocation          PSECT No.  Attributes
----------                       ----------          ---------  ----------
.  ABS  .                        00000000  (     0.)  00 (   0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
ECKAXA                           00000335  (   821.)  01 (   1.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD   NOWRT NOVEC PAGE
$ABS$                            00010610  (67088.)  02 (   2.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
TEST_001                         0000034D  (   845.)  03 (   3.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD   NOWRT NOVEC PAGE
DISPATCH                         00000018  (    24.)  04 (   4.)  NOPIC    USR    CON    REL    LCL NOSHR NOEXE   RD   NOWRT NOVEC LONG
ARGLIST                          00000030  (    48.)  05 (   5.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD   NOWRT NOVEC LONG
$TSTCNT                          00000004  (     4.)  06 (   6.)  NOPIC    USR    OVR    REL    LCL NOSHR NOEXE   RD   NOWRT NOVEC LONG
```

```
                                             +------------------------+
                                             ! Symbol Cross Reference !
                                             +------------------------+

SYMBOL                           VALUE       DEFINITION        REFERENCES...
------                           -----       ----------        -------------
$$$                             =00000024-R  497    (10)       294    (7)       367    (8)       432    (9)       497    (10)
$$ARGS                          =0000000A    99     (2)        99     (2)
$$E                             =00000001    497    (10)       324    (7)       390    (8)       460    (9)       518    (10)
$$M                             =00000004    515    (10)       254    (6)       305    (7)       313    (7)       378    (8)
                                                               384    (8)       444    (9)       450    (9)       505    (10)
                                                               511    (10)      515    (10)
$$N                             =00000000    515    (10)       100    (2)     #-221    (4)       222    (4)       305    (7)
                                                               313    (7)       378    (8)       384    (8)       444    (9)
                                                               450    (9)       505    (10)      511    (10)      515    (10)
$$S                             =FFFFFFFF    519    (10)       240    (6)       254    (6)       258    (7)       294    (7)
                                                               328    (8)       367    (8)       394    (9)       432    (9)
                                                               464    (10)      497    (10)
$$T1                            =0000002C    99     (2)        99     (2)
$ENV                            =00000001    73     (2)
$ER                             =FFFFFFFF    519    (10)       294    (7)     #-305    (7)     #-313    (7)       367    (8)
                                                             #-378    (8)     #-384    (8)       432    (9)     #-444    (9)
                                                             #-450    (9)       497    (10)    #-505    (10)    #-511    (10)
                                                             #-515    (10)
$MO                             =00000001    520    (10)       520    (10)
$S$                             =00000024-R  497    (10)       294    (7)       324    (7)       367    (8)       390    (8)
                                                               432    (9)       460    (9)       497    (10)      518    (10)
$ST                             =00000000    519    (10)       254    (6)       258    (7)       294    (7)       305    (7)
                                                               313    (7)       324    (7)       328    (8)       367    (8)
                                                               378    (8)       384    (8)       390    (8)       394    (9)
                                                               432    (9)       444    (9)       450    (9)       460    (9)
                                                               464    (10)      497    (10)      505    (10)      511    (10)
                                                               515    (10)      518    (10)      519    (10)
$TN                             =00000001    520    (10)       240    (6)       254    (6)       258    (7)       305    (7)
                                                               313    (7)       328    (8)       378    (8)       384    (8)
                                                               394    (9)       444    (9)       450    (9)       464    (10)
                                                               505    (10)      511    (10)      515    (10)      519    (10)
                                                               520    (10)
ADDRESS TABLE                    00000005-R  123    (3)        296    (7)
ALL                             =00000009    100    (2)        254    (6)
BASE 001                         00000000-R  240    (6)        254    (6)
BIT...                          =00000011    94     (2)         73    (2)        89    (2)        90    (2)        94    (2)
B_FLAGS                          00000004-R  107    (2)      #-235    (5)     #-298    (7)     #-302    (7)       304    (7)
                                                             #-307    (7)     #-310    (7)       312    (7)     #-373    (8)
                                                             #-375    (8)       377    (8)     #-380    (8)       383    (8)
                                                             #-439    (9)     #-441    (9)       443    (9)     #-446    (9)
                                                               449    (9)     #-500    (10)      504    (10)    #-507    (10)
                                                               510    (10)
CEP_FUNCTIONAL                  =00000000    73     (2)
CEP_REPAIR                      =00000001    73     (2)         73    (2)
DATA_001                         00000020-R  254    (6)        254    (6)
DEFAULT                         =00000000    100    (2)        254    (6)
DS$BGNSUB                        00010030                      294    (7)       367    (8)       432    (9)       497    (10)
DS$BREAK                         00010058                      519    (10)
```

```
DS$CLRVEC                         00010168                          322      (7)       389      (8)       459      (9)       516      (10)
DS$ENDSUB                         00010038                          324      (7)       390      (8)       460      (9)       518      (10)
DS$ERRHARD                        000100D0                          305      (7)       313      (7)       378      (8)       384      (8)
                                                                    444      (9)       450      (9)       505      (10)      511      (10)
                                                                    515      (10)
DS$K_ERROR                       =00000002      94       (2)
DS$K_NORMAL                      =00000001      94       (2)
DS$K_SEVERE                      =00000004      94       (2)
DS$K_SUBSYS                      =00000066      94       (2)        94       (2)
DS$K_WARNING                     =00000000      94       (2)
DS$PRINTB                         000100E0                          221      (4)       222      (4)
DS$SETVEC                         00010160                          295      (7)       369      (8)       433      (9)       435      (9)
                                                                    498      (10)
DS$_ARITH                        =006600D0      94       (2)
DS$_ASBE                         =00660118      94       (2)
DS$_BADLINK                      =006600F0      94       (2)
DS$_BADTYPE                      =006600E8      94       (2)
DS$_BIIC                         =00660120      94       (2)
DS$_CHME                         =006600A8      94       (2)
DS$_CHMK                         =006600E0      94       (2)
DS$_DEVNAME                      =00660108      94       (2)
DS$_ERROR                        =00660002      94       (2)
DS$_FHWE                         =00660068      94       (2)
DS$_FRAGBUF                      =00660080      94       (2)
DS$_ICBUSY                       =006600C8      94       (2)
DS$_ICERR                        =006600C0      94       (2)
DS$_IHWE                         =00660060      94       (2)
DS$_ILLCHAR                      =00660018      94       (2)
DS$_ILLPAGCNT                    =00660078      94       (2)
DS$_ILLUNIT                      =00660100      94       (2)
DS$_INSFMEM                      =00660050      94       (2)
DS$_IPL2HI                       =006600B8      94       (2)
DS$_IVADDR                       =00660040      94       (2)
DS$_IVVECT                       =00660038      94       (2)
DS$_KRNLSTK                      =00660090      94       (2)
DS$_LOGIC                        =00660070      94       (2)
DS$_MCHK                         =00660088      94       (2)
DS$_MMOFF                        =00660058      94       (2)
DS$_NEEDUNIT                     =006600F8      94       (2)
DS$_NODE                         =00660128      94       (2)
DS$_NOPCS                        =00660110      94       (2)
DS$_NORMAL                       =00660001      94       (2)
DS$_NOSUPPORT                    =006600B1      94       (2)
DS$_NOTDON                       =00660030      94       (2)
DS$_NOTIMP                       =006600B0      94       (2)        94       (2)
DS$_NULLSTR                      =00660010      94       (2)
DS$_OVERFLOW                     =00660008      94       (2)
DS$_POWER                        =00660098      94       (2)
DS$_PROGERR                      =00660020      94       (2)
DS$_SEVERE                       =00660004      94       (2)
DS$_TRANSL                       =006600A0      94       (2)
DS$_TRUNCATE                     =00660028      94       (2)
DS$_UNEXPINT                     =006600D8      94       (2)
DS$_VASFULL                      =00660048      94       (2)
DS$_WARNING                      =00660000      94       (2)        94       (2)
```

```
DSA$GL_SID                      0000FE14                   #-316    (7)     #-453    (9)
ENV$M_DOMAIN                   =00000002     73     (2)
ENV$M_LEVEL                    =00000001     73     (2)
ENV$M_SUPER                    =000003FC     73     (2)
ENV$S_DOMAIN                   =00000001     73     (2)
ENV$S_LEVEL                    =00000001     73     (2)
ENV$S_SUPER                    =00000008     73     (2)
ENV$V_DOMAIN                   =00000001     73     (2)       73    (2)
ENV$V_LEVEL                    =00000000     73     (2)       73    (2)
ENV$V_SUPER                    =00000002     73     (2)
ENV$_CPU                       =00000000     73     (2)       73    (2)
ENV$_FUNCTIONAL                =00000000     73     (2)       73    (2)
ENV$_REPAIR                    =00000001     73     (2)       73    (2)
ENV$_SUPER                     =00000001     73     (2)
ENV$_SYSTEM                    =00000001     73     (2)       73    (2)
ERR$_MSGADR                    =0000000C     99     (2)
ERR$_NARGS                     =0000000A     99     (2)
ERR$_NUM                       =00000004     99     (2)
ERR$_P1                        =00000014     99     (2)      221    (4)
ERR$_P2                        =00000018     99     (2)
ERR$_P3                        =0000001C     99     (2)
ERR$_P4                        =00000020     99     (2)
ERR$_P5                        =00000024     99     (2)
ERR$_P6                        =00000028     99     (2)
ERR$_POINTER                   =00000010     99     (2)
ERR$_UNIT                      =00000008     99     (2)
ERR_MESSAGE                     00000310-R   219    (4)      305    (7)       313    (7)       378    (8)       384    (8)
                                                             444    (9)       450    (9)       505    (10)      511    (10)
FAULT_SERVICE                   0000032C-R   233    (5)      295    (7)       369    (8)       433    (9)       435    (9)
                                                             498    (10)
HALT                           =00000004     100    (2)
IPR                            =00000006     100    (2)      254    (6)
KA$B_ACC_TYPE                   00000042                   #-513    (10)
KA_PTABLE                       00000000-XR                #-512    (10)
L_REG_ADD                       00000000-R   106    (2)    #-222    (4)     #-300    (7)     #-371    (8)     #-437    (9)
                                                           #-501    (10)
MANUAL                         =00000001     100    (2)
MCHK                           =00000005     100    (2)
MSG_001                         00000002-R   240    (6)      254    (6)
M_FAULT                        =00000002     112    (2)    #-235    (5)
M_NO_FAULT                     =00000001     110    (2)    #-302    (7)     #-310    (7)     #-375    (8)     #-441    (9)
NEXT                            00000049-R   297    (7)    #-321    (7)
NEXT1                           00000119-R   370    (8)    #-388    (8)
NEXT2                           000001E2-R   436    (9)    #-458    (9)
PAR$M_ATDEF                    =00000010     89     (2)
PAR$M_ATHI                     =00000008     89     (2)
PAR$M_ATLO                     =00000004     89     (2)
PAR$M_NODEF                    =00000002     89     (2)
PAR$V_ATDEF                    =00000004     89     (2)
PAR$V_ATHI                     =00000003     89     (2)
PAR$V_ATLO                     =00000002     89     (2)
PAR$V_NODEF                    =00000001     89     (2)
PAR$_BIN                       =00000002     89     (2)
PAR$_DEC                       =0000000A     89     (2)
PAR$_HEX                       =00000010     89     (2)
```

```
PAR$_NO                        =00000000     89     (2)
PAR$_OCT                       =00000008     89     (2)
PAR$_YES                       =00000001     89     (2)
PCS                            =00000008     100    (2)
POWER_FAIL                     =00000003     100    (2)
PR750$_ACCS                    =00000028                   #-502    (10)   #-508    (10)
PSL$K_EXEC                     =00000001     90     (2)
PSL$K_KERNEL                   =00000000     90     (2)
PSL$K_SUPER                    =00000002     90     (2)
PSL$K_USER                     =00000003     90     (2)
PSL$M_CBIT                     =00000001     90     (2)
PSL$M_NBIT                     =00000008     90     (2)
PSL$M_VBIT                     =00000002     90     (2)
PSL$M_ZBIT                     =00000004     90     (2)
PSL$V_CBIT                     =00000000     90     (2)
PSL$V_NBIT                     =00000003     90     (2)
PSL$V_VBIT                     =00000001     90     (2)
PSL$V_ZBIT                     =00000002     90     (2)
READ_TABLE                      000000D5-R   186    (3)      368    (8)
RENTRY_1                        0000006A-R   303    (7)      297    (7)
RENTRY_2                        000000A5-R   311    (7)      306    (7)
RENTRY_3                        0000013A-R   376    (8)      372    (8)
RENTRY_4                        0000016B-R   382    (8)      379    (8)
RENTRY_5                        00000203-R   442    (9)      438    (9)
RENTRY_6                        00000234-R   448    (9)      445    (9)
RENTRY_7                        000002BC-R   503    (10)     499    (10)
RENTRY_8                        000002ED-R   509    (10)     506    (10)
SCB$L_ROPRAND                   00000018                   #-295    (7)    #-322    (7)    #-369    (8)    #-389    (8)
                                                           #-433    (9)    #-435    (9)    #-459    (9)    #-498    (10)
                                                           #-516    (10)
SEP_FUNCTIONAL                 =00000002     73     (2)
SEP_REPAIR                     =00000003     73     (2)
SIZ...                         =00000001     90     (2)       73    (2)       89    (2)       90    (2)
T1_S1                           00000026-R   294    (7)
T1_S1_X                         000000EB-R   324    (7)
T1_S2                           000000F6-R   367    (8)
T1_S2_X                         000001A3-R   390    (8)
T1_S3                           000001AE-R   432    (9)
T1_S3_X                         0000027A-R   460    (9)
T1_S4                           00000285-R   497    (10)
T1_S4_X                         00000337-R   518    (10)
TEST_001                        00000024-R   254    (6)      254    (6)
TEST_001_X                      00000345-R   519    (10)
TU58                           =00000002     100    (2)
T_FAULT_ERR_F                   00000144-R   199    (4)    #-384    (8)    #-511    (10)
T_FAULT_ERR_T                   00000218-R   207    (4)    #-450    (9)    #-505    (10)
T_FPA_ERR                       00000283-R   211    (4)      515    (10)
T_NO_FAULT_FROM                 000001AF-R   203    (4)    #-313    (7)
T_NO_FAULT_TO                   000000DD-R   195    (4)    #-305    (7)    #-378    (8)    #-444    (9)
T_REG_ADD                       000002EA-R   215    (4)      222    (4)
V_FAULT                        =00000001     111    (2)    #-383    (8)    #-449    (9)    #-504    (10)   #-510    (10)
V_NO_FAULT                     =00000000     109    (2)    #-304    (7)    #-312    (7)    #-377    (8)    #-443    (9)
WCS                            =00000007     100    (2)
WRITE_TABLE                     000000C9-R   177    (3)      434    (9)
```

```
                                        +-------------------------+
                                        ! Macros Cross Reference !
                                        +-------------------------+

MACRO            SIZE         DEFINITION       REFERENCES...
-----            ----         ----------       -------------
$D1_BSUBT        1            294    (7)       294    (7)      367    (8)      432    (9)      497    (10)
$D1_BTEST        1            254    (6)       254    (6)
$D1_ERR          1            305    (7)       305    (7)      313    (7)      378    (8)      384    (8)      444    (9)
                                               450    (9)      505    (10)     511    (10)     515    (10)
$D1_ERR_S        1            305    (7)       305    (7)      313    (7)      378    (8)      384    (8)      444    (9)
                                               450    (9)      505    (10)     511    (10)     515    (10)
$D1_ESUBT        1            324    (7)       324    (7)      390    (8)      460    (9)      518    (10)
$D1_ETEST        1            519    (10)      519    (10)
$D1_PRINT_S      1            221    (4)       221    (4)      222    (4)
$D1_SBTTL        2            240    (6)       240    (6)      258    (7)      328    (8)      394    (9)      464    (10)
$D2_BDATA        1            254    (6)       254    (6)
$D2_BTEST        1            254    (6)       254    (6)
$D2_SBTTL        1            240    (6)       240    (6)
$DEF             1            94     (2)
$DEFINI          1            90     (2)       90     (2)      91     (2)      92     (2)      93     (2)      95     (2)
                                               96     (2)      97     (2)      98     (2)
$DS_BGNMESSAGE   1            220    (4)       220    (4)
$DS_BGNMOD       1            73     (2)       73     (2)
$DS_BGNSUB       1            294    (7)       294    (7)      367    (8)      432    (9)      497    (10)
$DS_BGNTEST      1            254    (6)       254    (6)
$DS_BREAK        1            519    (10)      519    (10)
$DS_CLRVEC_S     1            322    (7)       322    (7)      389    (8)      459    (9)      516    (10)
$DS_DSADEF       5            95     (2)       95     (2)
$DS_DSDEF        2            94     (2)       94     (2)
$DS_DSSDEF       8            96     (2)       96     (2)
$DS_ENDDATA      1            254    (6)       254    (6)
$DS_ENDMESSAGE   1            223    (4)       223    (4)
$DS_ENDMOD       1            520    (10)      520    (10)
$DS_ENDSUB       1            324    (7)       324    (7)      390    (8)      460    (9)      518    (10)
$DS_ENDTEST      1            519    (10)      519    (10)
$DS_ENVDEF       2            73     (2)       73     (2)
$DS_ERRDEF       1            99     (2)       99     (2)
$DS_ERRHARD_S    1            305    (7)       305    (7)      313    (7)      378    (8)      384    (8)      444    (9)
                                               450    (9)      505    (10)     511    (10)     515    (10)
$DS_HPODEF       2            97     (2)       97     (2)
$DS_PAGE         1            238    (5)       238    (5)      256    (6)      326    (7)      392    (8)      462    (9)
$DS_PARDEF       1            89     (2)       89     (2)
$DS_PRINTB_S     1            221    (4)       221    (4)      222    (4)
$DS_PSLDEF       1            90     (2)       90     (2)
$DS_SBTTL        1            240    (6)       240    (6)      258    (7)      328    (8)      394    (9)      464    (10)
$DS_SCBDEF       3            93     (2)       93     (2)
$DS_SECDEF       1            100    (2)       100    (2)
$DS_SETVEC_S     1            295    (7)       295    (7)      369    (8)      433    (9)      435    (9)      498    (10)
$EQU             1            94     (2)       73     (2)      89     (2)      90     (2)      94     (2)
$EQULS1          1            94     (2)       73     (2)      90     (2)      94     (2)
$EQULST          1            73     (2)       73     (2)      90     (2)      94     (2)
$GBLINI          2            73     (2)       73     (2)      89     (2)      90     (2)      94     (2)
$OFFDEF          1            99     (2)       99     (2)
```

```
$PR750DEF           1             92      (2)        92      (2)
$PRDEF              4             91      (2)        91      (2)
$PSLDEF             2             90      (2)        90      (2)
$PUSHADR            1             305     (7)        305     (7)       313     (7)       378     (8)       384     (8)       444     (9)
                                                     450     (9)       505     (10)      511     (10)      515     (10)
$VIELD              1             73      (2)        73      (2)       89      (2)       90      (2)
$VIELD1             1             94      (2)        73      (2)       89      (2)       90      (2)
ECKAX_SECDEF        1             100     (2)        100     (2)
KA_DEF              2             98      (2)        98      (2)

                                              +--------------------------+
                                              ! Performance indicators !
                                              +--------------------------+

Phase                         Page faults    CPU Time         Elapsed Time
-----                         -----------    --------         ------------
Initialization                       111     00:00:00.38      00:00:03.77
Command processing                   880     00:00:00.95      00:00:04.65
Pass 1                               558     00:00:16.03      00:00:37.62
Symbol table sort                      0     00:00:00.81      00:00:00.93
Pass 2                                 8     00:00:02.93      00:00:05.24
Symbol table output                    2     00:00:00.24      00:00:01.10
Psect synopsis output                  0     00:00:00.04      00:00:00.04
Cross-reference output                 6     00:00:01.03      00:00:03.19
Assembler run totals                1566     00:00:22.43      00:00:56.69

The working set limit was 2912 pages.
118261 bytes (231 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 564 non-local and 14 local symbols.
521 source lines were read in Pass 1, producing 25 object records in Pass 2.
125 pages of virtual memory were used to define 49 macros.

                                             +----------------------------+
                                             ! Macro library statistics !
                                             +----------------------------+

Macro library name                               Macros defined
------------------                               --------------
DISK$DIAGPACK02:[TAYLOR.ECKAX]ECKAX.MLB;1                  2
SYS$COMMON:[SYSLIB]DIAG.MLB;955                           37
SYS$COMMON:[SYSLIB]STARLET.MLB;1                          10
TOTALS (all libraries)                                    49

933 GETS were required to define 49 macros.

There were no errors, warnings or information messages.

MACRO/LIS=ECKAX2.LIS/CROSS ECKAX2
```

Table of contents

```
    (1)        1        ECKAX3: POWER FAIL
    (3)       67        DECLARATIONS
    (4)      131        INSTRUCTION MESSAGES
    (4)      189        ERROR MESSAGES
    (5)      235        SUBROUTINES
    (6)      359        TEST 2:   POWER FAIL (INIT SWITCH)
    (7)      393        SUBTEST 2.1:  GOOD RPB SUBTEST
    (8)      449        SUBTEST 2.2:  BAD CHECKSUM SUBTEST
    (9)      508        SUBTEST 2.3:  SEARCH FOR GOOD RPB
    (10)     570        SUBTEST 2.4:  WARM START FLAG SUBTEST
    (11)     628        TEST 3:   POWER FAIL (BREAKER)
    (12)     658        SUBTEST 3.1:  GOOD RPB SUBTEST
    (13)     715        SUBTEST 3.2:  BAD CHECKSUM SUBTEST
    (14)     775        SUBTEST 3.3:  SEARCH FOR GOOD RPB
    (15)     838        SUBTEST 3.4:  WARM START FLAG SUBTEST
```

```
0000             .SBTTL  ECKAX3: POWER FAIL
0000             .TITLE  ECKAX3  VAX-11/750 Specific CPU Cluster Exerciser
0000             .IDENT  /5.2/
0000
0000     ;
0000     ; Copyright (C) 1980,1983,1986
0000     ; Digital Equipment Corporation, Maynard, Massachusetts 01754
0000     ;
0000     ; This software is furnished  under a license for use only on a single
0000     ; computer  system and  may be  copied only with  the inclusion of the
0000     ; above copyright notice.  This software, or any other copies thereof,
0000     ; may not be provided or  otherwise made available to any other person
0000     ; except for use on such system and to one who agrees to these license
0000     ; terms.  Title to and  ownership of the  software  shall at all times
0000     ; remain in DEC.
0000     ;
0000     ; The information in this software is subject to change without notice
0000     ; and should  not be construed  as a commitment  by Digital  Equipment
0000     ; Corporation.
0000     ;
0000     ; DEC assumes  no  responsibility  for  the use or  reliability of its
0000     ; software on equipment which is not supplied by DEC.
0000     ;
0000
0000     ;++
0000     ; Facility:      VAX Architecture Diagnostic.
0000     ;
0000     ; Abstract:      This module tests that the VAX-11/750 power failure and
0000     ;                recovery mechanism functions properly.  The power fail
0000     ;                tests instruct the operator to select console pannel switch
0000     ;                settings and force various power interruptions.
0000     ;
0000     ;
0000     ; Environment:  VAX Diagnostic Supervisor.
0000     ;
0000     ; Author:        Fred Roemer      19-Feb-81        Version 2.0
0000     ;
0000     ;      NO CHANGES TO THIS MODULE FOR VERSION 3.3 (DOCUMENT CHANGE ONLY)
0000     ;
0000     ;
0000     ;      Dan Milleville  19-Mar-82               Version 3.4
0000     ;      Changed Test 3 exit code to accomodate temporarily a hardware problem.
0000     ;
0000     ;      Dave Shull      30-Nov-82               Version 4.0
0000     ;      Added tests 13 and 14 for coverage of PCS module.
0000     ;
0000     ;      Tai-Chun Pan    04-Mar-83               Version 5.0
0000     ;      Added test 5 for coverage of Processor Halts Architecture Undefined.
0000     ;      Added tests 11 and 12 for coverage of new ROM with MRSP of TU58.
0000     ;
0000     ;      Dan Milleville  21-Nov-83               Version 5.1
0000     ;      Fixed error in timer routine (didn't turn off timer).
0000     ;
0000     ;      Wayne R. Taylor February 1986           Version 5.2
0000     ;      Redefined IPR's to jive with starlet.
```

```
                                      0000     56 ;       Modified the code for problem report Eckax5 where:
                                      0000     57 ;               Tests 6 & 13 tests with cache off.
                                      0000     58 ;               Subtest 1 does not check bit 0 of the memory under test.
                                      0000     59 ;               Subtest 4 tests row 0, 1, 2, but not row 3.
                                      0000     60 ;       Indicate to user if the operator flag is not for tests 2 through 5.
                                      0000     61 ;       Modifications were supplied by the submitter.
                                      0000     62 ;       Search on 5.2.
                                      0000     63 ;
                                      0000     64 ;--
```

```
                 0000     66
                 0000     67            .SBTTL  DECLARATIONS
                 0000     68            .LIST   MEB
                 0000     69            .NLIST  CND
             00000000     70            .PSECT  ECKAXC, PAGE, NOWRT
                 0000     71            .LIBRARY        'SYS$LIBRARY:DIAG'; VAX family diagnostic library.
                 0000     72            $DS_BGNMOD      CEP_REPAIR, TN=2
                 0000        ;;;        Macro-32 Diagnostic macro library Version 8.3  DIAG.MLB (245)
                 0000     73
                 0000     74 ;
                 0000     75 ; Include files:
                 0000     76 ;
                 0000     77
                 0000     78            .LIBRARY         ECKAX          ; CPU Cluster Exerciser library.
                 0000     79
                 0000     80 ;
                 0000     81 ; Macros:
                 0000     82 ;
                 0000     83
                 0000     84            $DS_SCBDEF
                 0000     85            $DS_DSDEF
                 0000     86            $DS_DSSDEF
                 0000     87            $DS_PSLDEF
                 0000               .SAVE   LOCAL_BLOCK
                 0000               .PSECT  $ABS$,ABS
        00000000 0610               .=0
             00000000               .RESTORE
                 0000     88            $DS_PARDEF
                 0000     89            $DS_DSADEF
                 0000     90
                 0000     91            ECKAX_SECDEF
                 0000     92
                 0000     93
                 0000     94 ;
                 0000     95 ; Own storage:
                 0000     96 ;
                 0000     97
        00000004 0000     98 L_BUF_BEGIN:    .BLKL   1       ; address of data stored on power failure
        00000008 0004     99 L_COUNT:        .BLKL   1       ; counter
        0000000C 0008    100 L_CHECKSUM:     .BLKL   1       ; contains good RPB checksum
        00000010 000C    101 L_PSL:          .BLKL   1       ; used to store PSL
        00000014 0010    102 L_IS_SP:        .BLKL   1       ; used to store interrupt SP
        00000018 0014    103 L_KS_SP:        .BLKL   1       ; used to store kernel SP
                 0018    104
        00000019 0018    105 B_RESPONCE:     .BLKB   1       ; message responce byte
        0000001A 0019    106 B_FLAGS:        .BLKB   1       ; byte to keep power up and down done flags
        0000001B 001A    107 B_FLAGS2:       .BLKB   1       ; byte to keep stack and mode flags
                 001B    108
                 001B    109
        00000000 001B    110 V_PWRDN_DONE=0                  ; power down done position in flags
        00000001 001B    111 M_PWRDN_DONE=1@0                ; power down done mask in flags
        00000001 001B    112 V_PWRUP_DONE=1                  ; power up done position in flags
        00000002 001B    113 M_PWRUP_DONE=1@1                ; power up done mask in flags
        00000002 001B    114 V_BAD_PC=2                      ; incorrect pc position in flags
        00000004 001B    115 M_BAD_PC=1@2                    ; incorrect pc mask in flags
```

```
                      001B   116
          00000000    001B   117 V_BADMODE_DN=0                          ; bad mode on power down position in flags2
          00000001    001B   118 M_BADMODE_DN=1@0                        ; bad mode on power down mask in flags2
          00000001    001B   119 V_BADMODE_UP=1                          ; bad mode on power up position in flags2
          00000002    001B   120 M_BADMODE_UP=1@1                        ; bad mode on power up mask in flags2
          00000002    001B   121 V_BADSTACK_DN=2                         ; bad stack on power down position in flags2
          00000004    001B   122 M_BADSTACK_DN=1@2                       ; bad stack on power down mask in flags2
          00000003    001B   123 V_BADSTACK_UP=3                         ; bad stack on power up position in flags2
          00000008    001B   124 M_BADSTACK_UP=1@3                       ; bad stack on power up mask in flags2
                      001B   125
                      001B   126
  000009CC'9F   17    001B   127 JUMP:   JMP     @#POWER_UP              ; used to load instruction at loc 100 to
                      0021   128                                         ; restart from subtests which halt
          00000100    0021   129 STRTUP=^X100                            ; label for loc 100
```

```
                                     0021    131 .SBTTL  INSTRUCTION MESSAGES
                                     0021    132 ;--------------------------------------------------------------------------
                                     0021    133 T_INST:
6F 63 20 65 72 6F 66 65 42 2F 21 00' 0021    134          .ASCIC   !/Before continuing, set the front panel switch to HALT/RESTART!/
65 73 20 2C 67 6E 69 75 6E 69 74 6E  002D
20 74 6E 6F 72 66 20 65 68 74 20 74  0039
68 63 74 69 77 73 20 6C 65 6E 61 70  0045
53 45 52 2F 54 4C 41 48 20 6F 74 20  0051
               2F 21 54 52 41 54     005D
                                 41  0021
                                     0063    135
                                     0063    136 ;------------------------------------------------------------ ------- -----
                                     0063    137 T_MESSAGE_1:
74 73 65 52 20 64 6F 6F 47 2F 21 00' 0063    138          .ASCIC   !/Good Restart Parameter Block Subtest - This subtest /
65 74 65 6D 61 72 61 50 20 74 72 61  006F
74 62 75 53 20 6B 63 6F 6C 42 20 72  007B
73 20 73 69 68 54 20 2D 20 74 73 65  0087
6C 6C 69 77 2F 21 74 73 65 74 62 75  0093
73 74 69 20 74 72 61 74 73 65 72 20  009F
52 20 65 68 74 20 66 69 20 66 6C 65  00AB
72 65 70 6F 72 70 20 73 69 20 42 50  00B7
74 65 72 70 72 65 74 6E 69 20 79 6C  00C3
                  2F 21 2E 64 65     00CF
                                     00D4    139                   will restart itself if the RPB is properly interpreted.!/ -
69 6E 69 20 65 68 74 20 68 73 75 50  00D4    140                   Push the init switch to power fail...
20 6F 74 20 68 63 74 69 77 73 20 74  00E0
2E 2E 6C 69 61 66 20 72 65 77 6F 70  00EC
                                 2E  00F8
                                 95  0063
                                     00F9    141 ;--------------------------------------------------------------------------
                                     00F9    142 T_MESSAGE_2:
6B 63 65 68 43 20 64 61 42 2F 21 00' 00F9    143          .ASCIC   !/Bad Checksum Subtest - This subtest will halt the processor!/ -
20 74 73 65 74 62 75 53 20 6D 75 73  0105
65 74 62 75 73 20 73 69 68 54 20 2D  0111
74 6C 61 68 20 6C 6C 69 77 20 74 73  011D
73 73 65 63 6F 72 70 20 65 68 74 20  0129
                     2F 21 72 6F     0135
20 74 6C 61 68 20 61 20 68 74 69 77  0139    144                   with a halt code of 11. After this has happened the subtest!/ -
20 2E 31 31 20 66 6F 20 65 64 6F 63  0145
68 20 73 69 68 74 20 72 65 74 66 41  0151
20 64 65 6E 65 70 70 61 68 20 73 61  015D
21 74 73 65 74 62 75 73 20 65 68 74  0169
                                 2F  0175
64 65 74 72 61 74 73 65 72 20 73 69  0176    145                   is restarted by typing S 100 <CR> on the console terminal.!/ -
53 20 67 6E 69 70 79 74 20 79 62 20  0182
6E 6F 20 3E 52 43 3C 20 30 30 31 20  018E
65 6C 6F 73 6E 6F 63 20 65 68 74 20  019A
2F 21 2E 6C 61 6E 69 6D 72 65 74 20  01A6
69 6E 69 20 65 68 74 20 68 73 75 50  01B2    146                   Push the init switch to power fail...!/
20 6F 74 20 68 63 74 69 77 73 20 74  01BE
2E 2E 6C 69 61 66 20 72 65 77 6F 70  01CA
                           2F 21 2E  01D6
                                 DF  00F9
                                     01D9    147 ;--------------------------------------------------------------------------
                                     01D9    148 T_MESSAGE_3:
```

```
6F 66 20 68 63 72 61 65 53 2F 21 00' 01D9   149          .ASCIC   !/Search for Good RPB - This subtest will restart itself!/ -
2D 20 42 50 52 20 64 6F 6F 47 20 72  01E5
73 65 74 62 75 73 20 73 69 68 54 20  01F1
61 74 73 65 72 20 6C 6C 69 77 20 74  01FD
   2F 21 66 6C 65 73 74 69 20 74 72  0209
20 64 6F 6F 67 20 65 68 74 20 66 69  0214   150                   'if the good RPB is found and interpreted properly.!/ -
64 6E 75 6F 66 20 73 69 20 42 50 52  0220
72 70 72 65 74 6E 69 20 64 6E 61 20  022C
6C 72 65 70 6F 72 70 20 64 65 74 65  0238
                     2F 21 2E 79     0244
69 6E 69 20 65 68 74 20 68 73 75 50  0248   151                   Push the init switch to power fail...
20 6F 74 20 68 63 74 69 77 73 20 74  0254
2E 2E 6C 69 61 66 20 72 65 77 6F 70  0260
                                 2E  026C
                                 93  01D9
                                     026D   152 ;-------------------------------------------------------------------------
                                     026D   153 T_MESSAGE_4:
74 53 20 6D 72 61 57 2F 21 2F 21 00' 026D   154          .ASCIC   !/!/Warm Start Flag Subtest - This subtest will halt the processo -
cont> r!/'-
62 75 53 20 67 61 6C 46 20 74 72 61  0279
20 73 69 68 54 20 2D 20 74 73 65 74  0285
6C 6C 69 77 20 74 73 65 74 62 75 73  0291
72 70 20 65 68 74 20 74 6C 61 68 20  029D
         2F 21 72 6F 73 73 65 63 6F  02A9
20 74 6C 61 68 20 61 20 68 74 69 77  02B2   155                   with a halt code of 12. After this has happened the subtest!/ -
20 2E 32 31 20 66 6F 20 65 64 6F 63  02BE
68 20 73 69 68 74 20 72 65 74 66 41  02CA
20 64 65 6E 65 70 70 61 68 20 73 61  02D6
21 74 73 65 74 62 75 73 20 65 68 74  02E2
                                 2F  02EE
64 65 74 72 61 74 73 65 72 20 73 69  02EF   156                   is restarted by typing S 100 <CR> on the console terminal.!/ -
53 20 67 6E 69 70 79 74 20 79 62 20  02FB
6E 6F 20 3E 52 43 3C 20 30 30 31 20  0307
65 6C 6F 73 6E 6F 63 20 65 68 74 20  0313
2F 21 2E 6C 61 6E 69 6D 72 65 74 20  031F
69 6E 69 20 65 68 74 20 68 73 75 50  032B   157                   Push the init switch to power fail...!/
20 6F 74 20 68 63 74 69 77 73 20 74  0337
2E 2E 6C 69 61 66 20 72 65 77 6F 70  0343
                           2F 21 2E  034F
                                 E4  026D
                                     0352   158 ;------------------------------------------------------------------------
                                     0352   159 T_MESSAGE_5:
74 73 65 52 20 64 6F 6F 47 2F 21 00' 0352   160          .ASCIC   !/Good Restart Parameter Block Subtest - This subtest will!/ -
65 74 65 6D 61 72 61 50 20 74 72 61  035E
74 62 75 53 20 6B 63 6F 6C 42 20 72  036A
73 20 73 69 68 54 20 2D 20 74 73 65  0376
21 6C 6C 69 77 20 74 73 65 74 62 75  0382
                                 2F  038E
65 73 74 69 20 74 72 61 74 73 65 72  038F   161                   restart itself if the RPB is properly interpreted.!/'-
50 52 20 65 68 74 20 66 69 20 66 6C  039B
6C 72 65 70 6F 72 70 20 73 69 20 42  03A7
65 74 65 72 70 72 65 74 6E 69 20 79  03B3
                     2F 21 2E 64     03BF
72 62 20 65 68 74 20 77 6F 72 68 54  03C3   162                   Throw the breaker switch to power fail..."
68 63 74 69 77 73 20 72 65 6B 61 65  03CF
61 66 20 72 65 77 6F 70 20 6F 74 20  03DB
```

```
                     2E 2E 2E 6C 69  03E7
                                 99  0352
                                     03EC   163 ;--------------------------------------------------------------------------
                                     03EC   164 T_MESSAGE_6:
6B 63 65 68 43 20 64 61 42 2F 21 00' 03EC   165         .ASCIC  '!/Bad Checksum Subtest - This subtest will halt the processor!/ -
20 74 73 65 74 62 75 53 20 6D 75 73  03F8
65 74 62 75 73 20 73 69 68 54 20 2D  0404
74 6C 61 68 20 6C 6C 69 77 20 74 73  0410
73 73 65 63 6F 72 70 20 65 68 74 20  041C
                        2F 21 72 6F  0428
20 74 6C 61 68 20 61 20 68 74 69 77  042C   166                  with a halt code of 11. After this has happened the subtest!/ -
20 2E 31 31 20 66 6F 20 65 64 6F 63  0438
68 20 73 69 68 74 20 72 65 74 66 41  0444
20 64 65 6E 65 70 70 61 68 20 73 61  0450
21 74 73 65 74 62 75 73 20 65 68 74  045C
                                 2F  0468
64 65 74 72 61 74 73 65 72 20 73 69  0469   167                 'is restarted by typing S 100 <CR> on the console terminal.!/'-
53 20 67 6E 69 70 79 74 20 79 62 20  0475
6E 6F 20 3E 52 43 3C 20 30 30 31 20  0481
65 6C 6F 73 6E 6F 63 20 65 68 74 20  048D
2F 21 2E 6C 61 6E 69 6D 72 65 74 20  0499
72 62 20 65 68 74 20 77 6F 72 68 54  04A5   168                  Throw the breaker switch to power fail...!/
68 63 74 69 77 73 20 72 65 6B 61 65  04B1
61 66 20 72 65 77 6F 70 20 6F 74 20  04BD
               2F 21 2E 2E 2E 6C 69  04C9
                                 E3  03EC
                                     04D0   169 ;--------------------------------------------------------------------------
                                     04D0   170 T_MESSAGE_7:
6F 66 20 68 63 72 61 65 53 2F 21 00' 04D0   171         .ASCIC   !/Search for Good RPB - This subtest will restart itself!/"-
2D 20 42 50 52 20 64 6F 6F 47 20 72  04DC
73 65 74 62 75 73 20 73 69 68 54 20  04E8
61 74 73 65 72 20 6C 6C 69 77 20 74  04F4
   2F 21 66 6C 65 73 74 69 20 74 72  0500
20 64 6F 6F 67 20 65 68 74 20 66 69  050B   172                  if the good RPB is found and interpreted properly.!/ -
64 6E 75 6F 66 20 73 69 20 42 50 52  0517
72 70 72 65 74 6E 69 20 64 6E 61 20  0523
6C 72 65 70 6F 72 70 20 64 65 74 65  052F
                        2F 21 2E 79  053B
72 62 20 65 68 74 20 77 6F 72 68 54  053F   173                  Throw the breaker switch to power fail...'
68 63 74 69 77 73 20 72 65 6B 61 65  054B
61 66 20 72 65 77 6F 70 20 6F 74 20  0557
                     2E 2E 2E 6C 69  0563
                                 97  04D0
                                     0568   174 ;--------------------------------------------------------------------------
                                     0568   175 T_MESSAGE_8:
74 53 20 6D 72 61 57 2F 21 2F 21 00' 0568   176         .ASCIC   !/!/Warm Start Flag Subtest - This subtest will halt the processo -
cont> r!/ -
62 75 53 20 67 61 6C 46 20 74 72 61  0574
20 73 69 68 54 20 2D 20 74 73 65 74  0580
6C 6C 69 77 20 74 73 65 74 62 75 73  058C
72 70 20 65 68 74 20 74 6C 61 68 20  0598
         2F 21 72 6F 73 73 65 63 6F  05A4
20 74 6C 61 68 20 61 20 68 74 69 77  05AD   177                  with a halt code of 12. After this has happened the subtest!/ -
20 2E 32 31 20 66 6F 20 65 64 6F 63  05B9
68 20 73 69 68 74 20 72 65 74 66 41  05C5
20 64 65 6E 65 70 70 61 68 20 73 61  05D1
```

```
21 74 73 65 74 62 75 73 20 65 68 74  05DD
                                 2F  05E9
64 65 74 72 61 74 73 65 72 20 73 69  05EA   178                    'is restarted by typing S 100 <CR> on the console terminal.!/ -
53 20 67 6E 69 70 79 74 20 79 62 20  05F6
6E 6F 20 3E 52 43 3C 20 30 30 31 20  0602
65 6C 6F 73 6E 6F 63 20 65 68 74 20  060E
2F 21 2E 6C 61 6E 69 6D 72 65 74 20  061A
72 62 20 65 68 74 20 77 6F 72 68 54  0626   179                    'Throw the breaker switch to power fail...!/
68 63 74 69 77 73 20 72 65 6B 61 65  0632
61 66 20 72 65 77 6F 70 20 6F 74 20  063E
               2F 21 2E 2E 2E 6C 69  064A
                                 E8  0568
                                     0651   180 ;--------------------------------------------------------------------------
                                     0651   181 T_SWITCH:
74 73 65 72 20 65 72 6F 66 65 42 00' 0651   182         .ASCIC   Before restarting set front panel switch to the HALT position.!/
66 20 74 65 73 20 67 6E 69 74 72 61  065D
73 20 6C 65 6E 61 70 20 74 6E 6F 72  0669
65 68 74 20 6F 74 20 68 63 74 69 77  0675
69 74 69 73 6F 70 20 54 4C 41 48 20  0681
                  2F 21 2E 6E 6F     068D
                                 40  0651
                                     0692   183 ;--------------------------------------------------------------------------
                                     0692   184 T_BAT_BKUP:
73 20 73 69 68 74 20 73 49 0A 0D 00' 0692   185         .ASCIC  <13><10> Is this system equiped with a battery backup? <13><10>-
65 70 69 75 71 65 20 6D 65 74 73 79  069E
74 61 62 20 61 20 68 74 69 77 20 64  06AA
3F 70 75 6B 63 61 62 20 79 72 65 74  06B6
                                 0D  06C2
65 68 74 20 2C 6F 6E 20 66 49 28 0A  06C3   186                    (If no, then set the front panel switch to the HALT position.) <1 -
cont> 3><10>
72 66 20 65 68 74 20 74 65 73 20 6E  06CF
77 73 20 6C 65 6E 61 70 20 74 6E 6F  06DB
20 65 68 74 20 6F 74 20 68 63 74 69  06E7
6F 69 74 69 73 6F 70 20 54 4C 41 48  06F3
                  0A 0D 29 2E 6E     06FF
                                 71  0692
                                     0704   187 ;--------------------------------------------------------------------------
                                     0704   188
                                     0704   189 .SBTTL  ERROR MESSAGES
                                     0704   190 ;--------------------------------------------------------------------------
                                     0704   191 T_PWRUP_FAILURE:
74 6E 69 6F 70 20 6B 63 61 74 53 00' 0704   192         .ASCIC   Stack pointer not equal to expected SP after power up
6C 61 75 71 65 20 74 6F 6E 20 72 65  0710
64 65 74 63 65 70 78 65 20 6F 74 20  071C
6F 70 20 72 65 74 66 61 20 50 53 20  0728
               70 75 20 72 65 77     0734
                                 35  0704
                                     073A   193
                                     073A   194 ;--------------------------------------------------------------------------
                                     073A   195 ; 5.2
                                     073A   196 T_NO_OPER:
6C 61 75 6E 61 4D 20 2F 21 2F 21 00' 073A   197         .ASCIC   !/!/ Manual Section:  Tests 2 through 5 require operator interven -
cont> tion.!/'-
54 20 20 3A 6E 6F 69 74 63 65 53 20  0746
75 6F 72 68 74 20 32 20 73 74 73 65  0752
65 72 69 75 71 65 72 20 35 20 68 67  075E
6E 69 20 72 6F 74 61 72 65 70 6F 20  076A
```

```
21 2E 6E 6F 69 74 6E 65 76 72 65 74  0776
                                 2F  0782
72 65 70 6F 20 65 68 74 20 74 65 53  0783   198                     Set the operator flag to execute tests 2 through 5. !/ -
6F 74 20 67 61 6C 66 20 72 6F 74 61  078F
73 65 74 20 65 74 75 63 65 78 65 20  079B
68 67 75 6F 72 68 74 20 32 20 73 74  07A7
                  2F 21 20 2E 35 20  07B3
20 74 72 61 74 73 65 72 20 64 6E 61  07B9   199                    'and restart the sections. '
73 6E 6F 69 74 63 65 73 20 65 68 74  07C5
                              20 2E  07D1
                                 98  073A
                                     07D3   200
                                     07D3   201 ;----------------------------------------------------------------------
                                     07D3   202 T_PWRDN_ERROR:
20 6E 77 6F 64 20 72 65 77 6F 50 00' 07D3   203         .ASCIC   Power down sequence did not complete
64 69 64 20 65 63 6E 65 75 71 65 73  07DF
74 65 6C 70 6D 6F 63 20 74 6F 6E 20  07EB
                                 65  07F7
                                 24  07D3
                                     07F8   204
                                     07F8   205 ;------------------------------------------------------ ---------------
                                     07F8   206 T_PWRUP_ERROR:
65 73 20 70 75 20 72 65 77 6F 50 00' 07F8   207         .ASCIC  "Power up sequence did not complete
6E 20 64 69 64 20 65 63 6E 65 75 71  0804
   65 74 65 6C 70 6D 6F 63 20 74 6F  0810
                                 22  07F8
                                     081B   208
                                     081B   209 ;----------------------------------------------------------------------
                                     081B   210 T_BADMODE:
20 65 64 6F 6D 20 67 6E 6F 72 57 00' 081B   211         .ASCIC  'Wrong mode in PSL for power down and up
70 20 72 6F 66 20 4C 53 50 20 6E 69  0827
6E 61 20 6E 77 6F 64 20 72 65 77 6F  0833
                        70 75 20 64  083F
                                 27  081B
                                     0843   212
                                     0843   213 ;----------------------------------------------------------------------
                                     0843   214 T_BADMODE_DN:
20 65 64 6F 6D 20 67 6E 6F 72 57 00' 0843   215         .ASCIC   Wrong mode in PSL for power down sequence
70 20 72 6F 66 20 4C 53 50 20 6E 69  084F
65 73 20 6E 77 6F 64 20 72 65 77 6F  085B
               65 63 6E 65 75 71  0867
                                 29  0843
                                     086D   216
                                     086D   217 ;---------------------------------------------------------------------
                                     086D   218 T_BADMODE_UP:
20 65 64 6F 6D 20 67 6E 6F 72 57 00' 086D   219         .ASCIC  'Wrong mode in PSL for power up sequence
70 20 72 6F 66 20 4C 53 50 20 6E 69  0879
75 71 65 73 20 70 75 20 72 65 77 6F  0885
                        65 63 6E 65  0891
                                 27  086D
                                     0895   220
                                     0895   221 ;----------------------------------------------------  -
                                     0895   222 T_BADSTACK:
73 20 74 70 75 72 72 65 74 6E 49 00' 0895   223         .ASCIC   Interrupt stack not used for power down and up
65 73 75 20 74 6F 6E 20 6B 63 61 74  08A1
```

```
20 72 65 77 6F 70 20 72 6F 66 20 64  08AD
   70 75 20 64 6E 61 20 6E 77 6F 64  08B9
                                 2E  0895
                                     08C4     224
                                     08C4     225 ;---------------------------------------------------------------------------
                                     08C4     226 T_BADSTACK_DN:
73 20 74 70 75 72 72 65 74 6E 49 00' 08C4     227         .ASCIC  'Interrupt stack not used for power down sequence
65 73 75 20 74 6F 6E 20 6B 63 61 74  08D0
20 72 65 77 6F 70 20 72 6F 66 20 64  08DC
63 6E 65 75 71 65 73 20 6E 77 6F 64  08E8
                                 65  08F4
                                 30  08C4
                                     08F5     228
                                     08F5     229 ;---------------------------------------------------------------------------
                                     08F5     230 T_BADSTACK_UP:
73 20 74 70 75 72 72 65 74 6E 49 00' 08F5     231         .ASCIC  'Interrupt stack not used for power up sequence
65 73 75 20 74 6F 6E 20 6B 63 61 74  0901
20 72 65 77 6F 70 20 72 6F 66 20 64  090D
   65 63 6E 65 75 71 65 73 20 70 75  0919
                                 2E  08F5
                                     0924     232
                                     0924     233 ;---------------------------------------------------------------------------
```

```
                              0924    235 .SBTTL  SUBROUTINES
                              0924    236
                              0924    237 ; This is the Power Fail interrupt service routine which is entered
                              0924    238 ; upon an interrupt due to loss of power. The internal processor
                              0924    239 ; registers, frame pointer, stack pointer, argument pointer and
                              0924    240 ; all general purpose registers are saved in memory befor all power
                              0924    241 ; is lost. If this is completed sucessfully a flag is set and then
                              0924    242 ; this routine goes into a loop, waiting for power to be totaly lost.
                              0924    243
                              0924    244         .ALIGN  LONG
                              0924    245 POWER_DOWN:
        58   00000000'FF  DE  0924    246         MOVAL   @A_PWR_BUFFER,R8         ; load R8 with address of buffer
                F6DD CF   DC  092B    247         MOVPSL  L_PSL                    ; save contents of the current PSL
                02    18  EC  092F    248         CMPV    #PSL$V_CURMOD,#PSL$S_CURMOD,-
           00   F6D7 CF       0932    249                 L_PSL,#PSL$K_KERNEL      ; check PSL for current mode = kernel
                      05  13  0936    250         BEQL    1$                       ; br = kernel mode
           F6DD CF    01  88  0938    251         BISB2   #M_BADMODE_DN,B_FLAGS2   ; flag bad mode on power down
        15 F6CA CF    1A  E0  093D    252 1$:     BBS     #PSL$V_IS,L_PSL,2$       ; br = on interrupt stack
           F6D2 CF    04  88  0943    253         BISB2   #M_BADSTACK_DN,B_FLAGS2  ; flag bad stack on power down
   F6BF CF   00000000'8F  DB  0948    254         MFPR    #PR$_ISP,L_IS_SP         ; save interrupt stack pointer
           F6BE CF    5E  D0  0951    255         MOVL    SP,L_KS_SP               ; save kernel stack pointer
                      0E  11  0956    256         BRB     3$                       ; go save other processor registers
           F6B3 CF    5E  D0  0958    257 2$:     MOVL    SP,L_IS_SP               ; save interrupt stack pointer
   F6AE CF   00000000'8F  DB  095D    258         MFPR    #PR$_KSP,L_KS_SP         ; save kernel stack pointer
                88    5C  D0  0966    259 3$:     MOVL    AP,(R8)+                 ; save processor registers
                88    5D  D0  0969    260         MOVL    FP,(R8)+
        88   00000000'8F  DB  096C    261         MFPR    #PR$_ESP,(R8)+
        88   00000000'8F  DB  0973    262         MFPR    #PR$_SSP,(R8)+
        88   00000000'8F  DB  097A    263         MFPR    #PR$_USP,(R8)+
        88   00000000'8F  DB  0981    264         MFPR    #PR$_SBR,(R8)+
        88   00000000'8F  DB  0988    265         MFPR    #PR$_SLR,(R8)+
        88   00000000'8F  DB  098F    266         MFPR    #PR$_PCBB,(R8)+
        88   00000000'8F  DB  0996    267         MFPR    #PR$_P0BR,(R8)+
        88   00000000'8F  DB  099D    268         MFPR    #PR$_P0LR,(R8)+
        88   00000000'8F  DB  09A4    269         MFPR    #PR$_P1BR,(R8)+
        88   00000000'8F  DB  09AB    270         MFPR    #PR$_P1LR,(R8)+
        88   00000000'8F  DB  09B2    271         MFPR    #PR$_ASTLVL,(R8)+
        88   00000000'8F  DB  09B9    272         MFPR    #PR$_SCBB,(R8)+
           F63B CF    58  D0  09C0    273         MOVL    R8,L_BUF_BEGIN           ; save address of data in memory
           F64F CF    01  88  09C5    274         BISB2   #M_PWRDN_DONE,B_FLAGS    ; set done flag
                      FE  11  09CA    275 4$:     BRB     4$                       ; wait
                              09CC    276
                              09CC    277
                              09CC    278
                              09CC    279 ; This is the Power Up routine which is pointed to by either the Restart
                              09CC    280 ; Parameter Block (RPB) or is entered through the restart code when a
                              09CC    281 ; nonrecoverable power fail has occured. This routine will restore all
                              09CC    282 ; of the information which was saved in the power down routine and also
                              09CC    283 ; sets a done flag to indicate that the power up sequence was sucessful.
                              09CC    284
                              09CC    285 POWER_UP:
           58   F630 CF   D0  09CC    286         MOVL    L_BUF_BEGIN,R8           ; load address of data
        00000000'8F   78  DA  09D1    287         MTPR    -(R8),#PR$_SCBB          ; load SCBB to handle any exceptions
                F630 CF   DC  09D8    288         MOVPSL  L_PSL                    ; save current PSL
                02    18  EC  09DC    289         CMPV    #PSL$V_CURMOD,#PSL$S_CURMOD,-
```

```
          00    F62A CF        09DF   290             L_PSL,#PSL$K_KERNEL       ; check current mode = kernel
                     05   13   09E3   291        BEQL   1$                        ; br = current mode kernel
          F630 CF    02   88   09E5   292        BISB2  #M_BADMODE_UP,B_FLAGS2    ; flag bad mode on power up
       15 F61D CF    1A   E0   09EA   293 1$:    BBS    #PSL$V_IS,L_PSL,2$        ; br = on interrupt stack
          F625 CF    08   88   09F0   294        BISB2  #M_BADSTACK_UP,B_FLAGS2   ; flag bad stack on power up
   00000000'8F  F617 CF   DA   09F5   295        MTPR   L_IS_SP,#PR$_ISP          ; load interrupt SP
          5E    F612 CF   D0   09FE   296        MOVL   L_KS_SP,SP                ; load kernel SP
                     0E   11   0A03   297        BRB    3$                        ; restore other processor registers
          5E    F607 CF   D0   0A05   298 2$:    MOVL   L_IS_SP,SP                ; load interrupt SP
   00000000'8F  F606 CF   DA   0A0A   299        MTPR   L_KS_SP,#PR$_KSP          ; load kernel SP
        00000000'8F  78   DA   0A13   300 3$:    MTPR   -(R8),#PR$_ASTLVL         ; restore other processor registers
        00000000'8F  78   DA   0A1A   301        MTPR   -(R8),#PR$_P1LR
        00000000'8F  78   DA   0A21   302        MTPR   -(R8),#PR$_P1BR
        00000000'8F  78   DA   0A28   303        MTPR   -(R8),#PR$_P0LR
        00000000'8F  78   DA   0A2F   304        MTPR   -(R8),#PR$_P0BR
        00000000'8F  78   DA   0A36   305        MTPR   -(R8),#PR$_PCBB
        00000000'8F  78   DA   0A3D   306        MTPR   -(R8),#PR$_SLR
        00000000'8F  78   DA   0A44   307        MTPR   -(R8),#PR$_SBR
        00000000'8F  78   DA   0A4B   308        MTPR   -(R8),#PR$_USP
        00000000'8F  78   DA   0A52   309        MTPR   -(R8),#PR$_SSP
        00000000'8F  78   DA   0A59   310        MTPR   -(R8),#PR$_ESP
                5D   78   D0   0A60   311        MOVL   -(R8),FP
                5C   78   D0   0A63   312        MOVL   -(R8),AP
       1E F5AF CF    00   E0   0A66   313        BBS    #V_BADMODE_DN,B_FLAGS2,5$ ; br = bad mode on power down
       4E F5A9 CF    01   E1   0A6C   314        BBC    #V_BADMODE_UP,B_FLAGS2,9$ ; br = both modes good
                               0A72   315        $DS_ERRHARD_S    UNIT=#0, MSGADR=T_BADMODE_UP
                     00   DD   0A72                   PUSHL    #0
                FDF5 CF   DF   0A74                   PUSHAL   T_BADMODE_UP
                     00   DD   0A78                   PUSHL    #0
                     01   DD   0A7A                   PUSHL    #$ER
                               0A7C          ;;; TEST 2, SUBTEST 0, ERROR 1
        000100D0 9F  04   FB   0A7C                   CALLS    #$$M, @#DS$ERRHARD
                               0A83   316        $DS_ABORT
        00010020 9F  6C   FA   0A83                   CALLG    (AP), @#DS$ABORT
       18 F58B CF    01   E1   0A8A   317 5$:    BBC    #V_BADMODE_UP,B_FLAGS2,7$ ; br = bad mode on power down only
                               0A90   318        $DS_ERRHARD_S    UNIT=#0, MSGADR=T_BADMODE
                     00   DD   0A90                   PUSHL    #0
                FD85 CF   DF   0A92                   PUSHAL   T_BADMODE
                     00   DD   0A96                   PUSHL    #0
                     02   DD   0A98                   PUSHL    #$ER
                               0A9A          ;;; TEST 2, SUBTEST 0, ERROR 2
        000100D0 9F  04   FB   0A9A                   CALLS    #$$M, @#DS$ERRHARD
                               0AA1   319        $DS_ABORT
        00010020 9F  6C   FA   0AA1                   CALLG    (AP), @#DS$ABORT
                               0AA8   320 7$:    $DS_ERRHARD_S    UNIT=#0, MSGADR=T_BADMODE_DN
                     00   DD   0AA8                   PUSHL    #0
                FD95 CF   DF   0AAA                   PUSHAL   T_BADMODE_DN
                     00   DD   0AAE                   PUSHL    #0
                     03   DD   0AB0                   PUSHL    #$ER
                               0AB2          ;;; TEST 2, SUBTEST 0, ERROR 3
        000100D0 9F  04   FB   0AB2                   CALLS    #$$M, @#DS$ERRHARD
                               0AB9   321        $DS_ABORT
        00010020 9F  6C   FA   0AB9                   CALLG    (AP), @#DS$ABORT
       1E F555 CF    02   E0   0AC0   322 9$:    BBS    #V_BADSTACK_DN,B_FLAGS2,11$ ; br = bad stack on power down
       4E F54F CF    03   E1   0AC6   323        BBC    #V_BADSTACK_UP,B_FLAGS2,15$ ; br = both stacks good
```

```
                                     0ACC   324         $DS_ERRHARD_S   UNIT=#0, MSGADR=T_BADSTACK_UP
                            00   DD  0ACC                       PUSHL   #0
                       FE23 CF   DF  0ACE                       PUSHAL  T_BADSTACK_UP
                            00   DD  0AD2                       PUSHL   #0
                            04   DD  0AD4                       PUSHL   #$ER
                                     0AD6       ;;; TEST 2, SUBTEST 0, ERROR 4
              000100D0 9F   04   FB  0AD6                       CALLS   #$$M, @#DS$ERRHARD
                                     0ADD   325         $DS_ABORT
              00010020 9F   6C   FA  0ADD                       CALLG   (AP), @#DS$ABORT
               18 F531 CF   03   E1  0AE4   326 11$:    BBC     #V_BADSTACK_UP,B_FLAGS2,13$ ; br = bad stack power down only
                                     0AEA   327         $DS_ERRHARD_S   UNIT=#0, MSGADR=T_BADSTACK
                            00   DD  0AEA                       PUSHL   #0
                       FDA5 CF   DF  0AEC                       PUSHAL  T_BADSTACK
                            00   DD  0AF0                       PUSHL   #0
                            05   DD  0AF2                       PUSHL   #$ER
                                     0AF4       ;;; TEST 2, SUBTEST 0, ERROR 5
              000100D0 9F   04   FB  0AF4                       CALLS   #$$M, @#DS$ERRHARD
                                     0AFB   328         $DS_ABORT
              00010020 9F   6C   FA  0AFB                       CALLG   (AP), @#DS$ABORT
                                     0B02   329 13$:    $DS_ERRHARD_S   UNIT=#0, MSGADR=T_BADSTACK_DN
                            00   DD  0B02                       PUSHL   #0
                       FDBC CF   DF  0B04                       PUSHAL  T_BADSTACK_DN
                            00   DD  0B08                       PUSHL   #0
                            06   DD  0B0A                       PUSHL   #$ER
                                     0B0C       ;;; TEST 2, SUBTEST 0, ERROR 6
              000100D0 9F   04   FB  0B0C                       CALLS   #$$M, @#DS$ERRHARD
                                     0B13   330         $DS_ABORT
              00010020 9F   6C   FA  0B13                       CALLG   (AP), @#DS$ABORT
                  F4FA CF   02   88  0B1A   331 15$:    BISB2   #M_PWRUP_DONE,B_FLAGS   ; set done flag
                                 02  0B1F   332         REI                             ; return to test
                                     0B20   333
                                     0B20   334
                                     0B20   335 ; This routine will calculate the checksum for the power up routine to
                                     0B20   336 ; be stored in the restart paramerter block.
                                     0B20   337
                                     0B20   338 CALC_CHECKSUM:
                            54   D4  0B20   339         CLRL    R4                      ; init R4 to store checksum
                  F4DD CF   01   D0  0B22   340         MOVL    #1,L_COUNT              ; initialize counter
                  53   FEA1 CF   DE  0B27   341         MOVAL   POWER_UP,R3             ; get address of power up routine
                       54   83   C0  0B2C   342 1$:     ADDL2   (R3)+,R4                ; add long word checksum
               F7 F4D0 CF   1F   F3  0B2F   343         AOBLEQ  #^X1F,L_COUNT,1$        ; br = not finished
                  F4CE CF   54   D0  0B35   344         MOVL    R4,L_CHECKSUM           ; save checksum
                                 05  0B3A   345         RSB                             ; return
                                     0B3B   346
                                     0B3B   347 ; This routine will clear the first page of buffer area.
                                     0B3B   348
                                     0B3B   349 CLEAR_BUFFER:
              55   00000000'FF   DE  0B3B   350         MOVAL   @A_BUFFER_BEGIN,R5      ; get starting buffer address
                       F4BE CF   D4  0B42   351         CLRL    L_COUNT                 ; init counter
                            85   D4  0B46   352 1$:     CLRL    (R5)+                   ; clear buffer location
      F4 F4B3 CF   00000080 8F   F3  0B48   353         AOBLEQ  #128,L_COUNT,1$         ; br = not finished
                                 05  0B52   354         RSB                             ; return
                                     0B53   355
                                     0B53   356
                                     0B53   357         $DS_PAGE
```

```
                              0B53                 .SBTTL  TEST 2:   POWER FAIL (INIT SWITCH)
                          00000000                         .PSECT  TEST_002, PAGE, NOWRT
                              0020
                              0020   360 ;++
                              0020   361 ; Test description:
                              0020   362 ;
                              0020   363 ;       This test consists of a series of manual operator intervention
                              0020   364 ;       subtests. These subtests will check that the Restart Parameter Block
                              0020   365 ;       (RPB) is recognizable and causes the proper sequence of events in
                              0020   366 ;       the occurrence of a power failure. Initially the power fail is
                              0020   367 ;       induced by pushing the initialize switch on the front panel of the
                              0020   368 ;       11/750. This simulates a power fail without removing all power from
                              0020   369 ;       the system. When this test is complete the operator will be prompted
                              0020   370 ;       to repeat the test using the breaker switch on the back of the system
                              0020   371 ;       if it is equipped with a battery backup. If a negative responce is
                              0020   372 ;       received the test is exited and the power fail test is complete.
                              0020   373 ;
                              0020   374 ;
                              0020   375 ; Assumptions:
                              0020   376 ;
                              0020   377 ;       For this test to function properly the right front panel select
                              0020   378 ;       switch must be in the HALT/RESTART position and the key switch
                              0020   379 ;       must be in the LOCAL position.
                              0020   380 ;
                              0020   381 ;--
                              0020   382
                              0020   383         $DS_BGNTEST     <MANUAL,POWER_FAIL,ALL>
                              0020       DATA_002:
                  00000000    0020                 .LONG   0               ; TEST ARGUMENT TABLE TERMINATOR
                              0024       TEST_002::
                      0000    0024                 .WORD   ^M<>            ; ENTRY MASK
                              0026   384         $DS_BOPER  1$                       ; Is operator present?
                   0C     E0  0026                 BBS     #DSA$V_OPER,-   ; BR IF OPERATOR
       14 0000FE00 9F         0028                         @#DSA$GL_FLAGS, 1$
                              002E   385         $DS_PRINTF_S    FORMAT=T_NO_OPER        ; Indicate no operator. 5.2
          0000073A'EF     9F  002E                 PUSHAB  T_NO_OPER
      000100F0 9F  01     FB  0034                 CALLS   #$$N, @#DS$PRINTF
                              003B   386         $DS_ABORT                           ; No, Abort
      00010020 9F  6C     FA  003B                 CALLG   (AP), @#DS$ABORT
                              0042   387 1$:     $DS_SETVEC_S    VECTOR=#SCB$L_POWER, SRVADR=POWER_DOWN, CODE=#1
                   01     DD  0042                 PUSHL   #1
          00000924'EF     DF  0044                 PUSHAL  POWER_DOWN
                   0C     DD  004A                 PUSHL   #SCB$L_POWER
      00010160 9F  03     FB  004C                 CALLS   #3, @#DS$SETVEC
                              0053   388
          00000B20'EF     16  0053   389 2$:     JSB     CALC_CHECKSUM   ; calculate good checksum for RPB.
                              0059   390         $DS_PRINTF_S    FORMAT=T_INST
          00000021'EF     9F  0059                 PUSHAB  T_INST
      000100F0 9F  01     FB  005F                 CALLS   #$$N, @#DS$PRINTF
                              0066   391         $DS_PAGE
```

```
                                     0066                 .SBTTL  SUBTEST 2.1:  GOOD RPB SUBTEST
                                     0066  394 ;+
                                     0066  395 ; SUBTEST DESCRIPTION:
                                     0066  396 ;
                                     0066  397 ;        This subtest sets up a good RPB at location zero, which points to
                                     0066  398 ;        a power up routine that restores all pertinent processor registers,
                                     0066  399 ;        the argument pointer, frame pointer and stack pointer. The operator
                                     0066  400 ;        is instructed through the console terminal to push the initialize
                                     0066  401 ;        switch on the front panel to simulate a CPU power failure. If the
                                     0066  402 ;        RPB is intrepreted properly the instrucions for the next subtest will
                                     0066  403 ;        be printed on the console terminal. If an error occured, either the
                                     0066  404 ;        processor will halt unexpectedly or an error report will be generated
                                     0066  405 ;        by the diagnostic supervisor.
                                     0066  406 ;
                                     0066  407 ;
                                     0066  408 ; SUBTEST STEPS:
                                     0066  409 ;
                                     0066  410 ;        SUBTEST GOOD RPB
                                     0066  411 ;        :  Setup good RPB at location 0
                                     0066  412 ;        :  Send instructions to operator
                                     0066  413 ;        :  Recover from power fail
                                     0066  414 ;        ENDSUBTEST GOOD RPB
                                     0066  415 ;
                                     0066  416 ;
                                     0066  417 ; ERRORS:
                                     0066  418 ;
                                     0066  419 ;
                                     0066  420 ; DEBUG:
                                     0066  421 ;
                                     0066  422 ;
                                     0066  423 ;-
                                     0066  424              $DS_BGNSUB
                                     0066      T2_S1::
     00010030 9F   00000000'EF   FA  0066                         CALLG   $S$, @#DS$BGNSUB
                                     0071  425
                   00000B3B'EF   16  0071  426          JSB     CLEAR_BUFFER    ; clear buffer area
                   00000019'EF   94  0077  427          CLRB    B_FLAGS         ; initialize flags
                   0000001A'EF   94  007D  428          CLRB    B_FLAGS2        ; init flags2
                       55   00   D0  0083  429          MOVL    #0,R5           ; load address to build RPB
                       85   00   D0  0086  430          MOVL    #0,(R5)+        ; load first longword with the address
              85   000009CC'EF   DE  0089  431          MOVAL   POWER_UP,(R5)+  ; load address of restart routine
              85   00000008'EF   D0  0090  432          MOVL    L_CHECKSUM,(R5)+ ; load good restart routine checksum
                            65   D4  0097  433          CLRL    (R5)            ; clear warm start flag
                                     0099  434          $DS_PRINTF_S    FORMAT=T_MESSAGE_1
                   00000063'EF   9F  0099                       PUSHAB  T_MESSAGE_1
              000100F0 9F   01   FB  009F                       CALLS   #$$N, @#DS$PRINTF
                                     00A6  435
                                     00A6  436 POWER_DOWN_1:
                                     00A6  437          $DS_BREAK
              00010058 9F   6E   FA  00A6                       CALLG   (SP), @#DS$BREAK
           02 00000019'EF   01   E0  00AD  438          BBS     #V_PWRUP_DONE,B_FLAGS,1$ ; br = power up done
                            EF   11  00B5  439          BRB     POWER_DOWN_1    ; wait for power fail interrupt
                                     00B7  440
           1A 00000019'EF   00   E0  00B7  441 1$:      BBS     #V_PWRDN_DONE,B_FLAGS,3$ ; br = power down sequence complete
                                     00BF  442          $DS_ERRHARD_S   UNIT=#0, MSGADR=T_PWRDN_ERROR
                       00    DD  00BF                          PUSHL    #0
              000007D3'EF    DF  00C1                          PUSHAL   T_PWRDN_ERROR
                       00    DD  00C7                          PUSHL    #0
                       01    DD  00C9                          PUSHL    #$ER
                                 00CB        ;;; TEST 2, SUBTEST 1, ERROR 1
          000100D0 9F  04    FB  00CB                          CALLS    #$$M, @#DS$ERRHARD
                                 00D2   443          $DS_ABORT
          00010020 9F  6C    FA  00D2                          CALLG    (AP), @#DS$ABORT
                                 00D9   444 3$:
                                 00D9   445          $DS_ENDSUB
                                 00D9        T2_S1_X:
 00010038 9F  00000000'EF    FA  00D9                          CALLG    $S$, @#DS$ENDSUB
                                 00E4   446
                                 00E4   447          $DS_PAGE
```

```
                                     00E4               .SBTTL  SUBTEST 2.2:  BAD CHECKSUM SUBTEST
                                     00E4   450 ;+
                                     00E4   451 ; SUBTEST DESCRIPTION:
                                     00E4   452 ;
                                     00E4   453 ;       This subtest sets up an RPB at location zero with a bad checksum.
                                     00E4   454 ;       It points to a power up routine that restores all pertinent processor
                                     00E4   455 ;       registers, the argument pointer, frame pointer and stack pointer.
                                     00E4   456 ;       The operator is instructed through the console terminal to push the
                                     00E4   457 ;       initialize switch on the front panel to simulate a CPU power failure,
                                     00E4   458 ;       check the halt code, and start a the indicated location. If the RPB is
                                     00E4   459 ;       intrepreted properly a halt code of 11 will be printed on the console
                                     00E4   460 ;       terminal. At this point the machine will be in console mode and the
                                     00E4   461 ;       operator should type  S 100' followed by a carrage return to continue
                                     00E4   462 ;       testing. If the expected halt code is not printed on the console
                                     00E4   463 ;       terminal then an error has occured.
                                     00E4   464 ;
                                     00E4   465 ;
                                     00E4   466 ; SUBTEST STEPS:
                                     00E4   467 ;
                                     00E4   468 ;       SUBTEST BAD CHECKSUM
                                     00E4   469 ;       :  Setup RPB at location 0 with bad checksum
                                     00E4   470 ;       :  Send instructions to operator
                                     00E4   471 ;       :  Recover from fatal power fail
                                     00E4   472 ;       ENDSUBTEST BAD CHECKSUM
                                     00E4   473 ;
                                     00E4   474 ;
                                     00E4   475 ; ERRORS:
                                     00E4   476 ;
                                     00E4   477 ;
                                     00E4   478 ; DEBUG:
                                     00E4   479 ;
                                     00E4   480 ;
                                     00E4   481 ;-
                                     00E4   482         $DS_BGNSUB
                                     00E4       T2_S2::
     00010030 9F   0000000C'EF   FA  00E4                       CALLG   $S$, @#DS$BGNSUB
                   00000B3B'EF   16  00EF   483         JSB     CLEAR_BUFFER    ; clear buffer area
                   00000019'EF   94  00F5   484         CLRB    B_FLAGS         ; initialize flags
                   0000001A'EF   94  00FB   485         CLRB    B_FLAGS2        ; init flags2
                       55   00   D0  0101   486         MOVL    #0,R5           ; load address to build RPB
                       85   00   D0  0104   487         MOVL    #0,(R5)+        ; load first longword with the address
              85   000009CC'EF   DE  0107   488         MOVAL   POWER_UP,(R5)+  ; load address of restart routine
              65   00000008'EF   D0  010E   489         MOVL    L_CHECKSUM,(R5) ; load good restart routine checksum
                       85   01   C2  0115   490         SUBL2   #1,(R5)+        ; make the checksum bad
                            65   D4  0118   491         CLRL    (R5)            ; clear warm start flag
     00000100'EF   0000001B'EF   7D  011A   492         MOVQ    JUMP,STRTUP     ; load addr 100 with JMP @#POWER_UP to restart
                                     0125   493         $DS_PRINTF_S    FORMAT=T_MESSAGE_2
                   000000F9'EF   9F  0125                       PUSHAB  T_MESSAGE_2
              000100F0 9F   01   FB  012B                       CALLS   #$$N, @#DS$PRINTF
                                     0132   494
                                     0132   495 POWER_DOWN_2:
                                     0132   496         $DS_BREAK
              00010058 9F   6E   FA  0132                       CALLG   (SP), @#DS$BREAK
           02 00000019'EF   01   E0  0139   497         BBS     #V_PWRUP_DONE,B_FLAGS,1$ ; br = power up sequence complete
                            EF   11  0141   498         BRB     POWER_DOWN_2    ; wait for power fail interrupt
```

```
                                  0143     499
       1A 00000019'EF    00    E0  0143     500 1$:     BBS     #V_PWRDN_DONE,B_FLAGS,3$ ; br = power down sequence complete
                                  014B     501         $DS_ERRHARD_S   UNIT=#0, MSGADR=T_PWRDN_ERROR
                         00    DD  014B                         PUSHL   #0
              000007D3'EF      DF  014D                         PUSHAL  T_PWRDN_ERROR
                         00    DD  0153                         PUSHL   #0
                         01    DD  0155                         PUSHL   #$ER
                                  0157         ;;; TEST 2, SUBTEST 2, ERROR 1
          000100D0 9F    04    FB  0157                         CALLS   #$$M, @#DS$ERRHARD
                                  015E     502         $DS_ABORT
          00010020 9F    6C    FA  015E                         CALLG   (AP), @#DS$ABORT
                                  0165     503 3$:
                                  0165     504         $DS_ENDSUB
                                  0165         T2_S2_X:
    00010038 9F   0000000C'EF    FA  0165                         CALLG   $S$, @#DS$ENDSUB
                                  0170     505
                                  0170     506         $DS_PAGE
```

```
                                     0170               .SBTTL  SUBTEST 2.3:  SEARCH FOR GOOD RPB
                                     0170   509 ;+
                                     0170   510 ; SUBTEST DESCRIPTION:
                                     0170   511 ;
                                     0170   512 ;       This subtest sets up an RPB at location zero with a bad checksum.
                                     0170   513 ;       It also sets up a good RPB at a higher page boundary which points
                                     0170   514 ;       to a power up routine that restores all pertinent processor
                                     0170   515 ;       registers, the argument pointer, frame pointer and stack pointer.
                                     0170   516 ;       The operator is instructed through the console terminal to push the
                                     0170   517 ;       initialize switch on the front panel to simulate a CPU power failure.
                                     0170   518 ;       If the RPB is intrepreted properly the next subtest instructions will
                                     0170   519 ;       be printed on the console terminal. If an error occured, either the
                                     0170   520 ;       processor will halt unexpectedly or an error report will be generated
                                     0170   521 ;       by the diagnostic supervisor.
                                     0170   522 ;
                                     0170   523 ;
                                     0170   524 ; SUBTEST STEPS:
                                     0170   525 ;
                                     0170   526 ;       SUBTEST SEARCH FOR GOOD RPB
                                     0170   527 ;       :  Setup RPB at location 0 with bad checksum
                                     0170   528 ;       :  Setup good RPB at higher page boundary
                                     0170   529 ;       :  Send instructions to operator
                                     0170   530 ;       ENDSUBTEST SEARCH FOR GOOD RPB
                                     0170   531 ;
                                     0170   532 ;
                                     0170   533 ; ERRORS:
                                     0170   534 ;
                                     0170   535 ;
                                     0170   536 ; DEBUG:
                                     0170   537 ;
                                     0170   538 ;
                                     0170   539 ;-
                                     0170   540         $DS_BGNSUB
                                     0170       T2_S3::
     00010030 9F   00000018'EF   FA  0170                       CALLG   $S$, @#DS$BGNSUB
                   00000B3B'EF   16  017B   541         JSB     CLEAR_BUFFER     ; clear buffer area
                   00000019'EF   94  0181   542         CLRB    B_FLAGS          ; initialize flags
                   0000001A'EF   94  0187   543         CLRB    B_FLAGS2         ; init flags2
                       55    00  D0  018D   544         MOVL    #0,R5            ; load address to build bad RPB
                       85    00  D0  0190   545         MOVL    #0,(R5)+         ; load first longword with the address
              85   000009CC'EF   DE  0193   546         MOVAL   POWER_UP,(R5)+   ; load address of restart routine
              65   00000008'EF   D0  019A   547         MOVL    L_CHECKSUM,(R5)  ; load good restart routine checksum
                       85    01  C0  01A1   548         ADDL2   #1,(R5)+         ; make checksum bad
                             65  D4  01A4   549         CLRL    (R5)             ; clear warm start flag
              55   00000000'FF   DE  01A6   550         MOVAL   @A_BUFFER_BEGIN,R5 ; load address to build good RPB
              85   00000000'FF   DE  01AD   551         MOVAL   @A_BUFFER_BEGIN,(R5)+ ; load first longword with the address
              85   000009CC'EF   DE  01B4   552         MOVAL   POWER_UP,(R5)+   ; load address of restart routine
              85   00000008'EF   D0  01BB   553         MOVL    L_CHECKSUM,(R5)+ ; load good restart routine checksum
                             65  D4  01C2   554         CLRL    (R5)             ; clear warm start flag
                                     01C4   555         $DS_PRINTF_S    FORMAT=T_MESSAGE_3
                   000001D9'EF   9F  01C4                       PUSHAB  T_MESSAGE_3
              000100F0 9F    01  FB  01CA                       CALLS   #$$N, @#DS$PRINTF
                                     01D1   556
                                     01D1   557 POWER_DOWN_3:
                                     01D1   558         $DS_BREAK
```

```
           00010058 9F    6E    FA  01D1                        CALLG   (SP), @#DS$BREAK
        02 00000019'EF    01    E0  01D8    559         BBS     #V_PWRUP_DONE,B_FLAGS,1$ ; br = power up sequence complete
                          EF    11  01E0    560         BRB     POWER_DOWN_3    ; wait for power fail interrupt
                                    01E2    561
        1A 00000019'EF    00    E0  01E2    562 1$:     BBS     #V_PWRDN_DONE,B_FLAGS,3$ ; br = power down sequence complete
                                    01EA    563         $DS_ERRHARD_S   UNIT=#0, MSGADR=T_PWRDN_ERROR
                          00    DD  01EA                        PUSHL   #0
                000007D3'EF     DF  01EC                        PUSHAL  T_PWRDN_ERROR
                          00    DD  01F2                        PUSHL   #0
                          01    DD  01F4                        PUSHL   #$ER
                                    01F6        ;;; TEST 2, SUBTEST 3, ERROR 1
           000100D0 9F    04    FB  01F6                        CALLS   #$$M, @#DS$ERRHARD
                                    01FD    564         $DS_ABORT
           00010020 9F    6C    FA  01FD                        CALLG   (AP), @#DS$ABORT
                                    0204    565 3$:
                                    0204    566         $DS_ENDSUB
                                    0204        T2_S3_X:
  00010038 9F     00000018'EF   FA  0204                        CALLG   $S$, @#DS$ENDSUB
                                    020F    567
                                    020F    568         $DS_PAGE
```

```
                                     020F               .SBTTL  SUBTEST 2.4:  WARM START FLAG SUBTEST
                                     020F   571 ;+
                                     020F   572 ; SUBTEST DESCRIPTION:
                                     020F   573 ;
                                     020F   574 ;       This subtest sets up an RPB at location zero with the warm start flag
                                     020F   575 ;       set. The RPB points to a power up routine that restores all pertinent
                                     020F   576 ;       processor registers, the argument pointer, frame pointer and stack
                                     020F   577 ;       pointer. The operator is instructed through the console terminal to
                                     020F   578 ;       push the initialize switch on the front panel to simulate a CPU power
                                     020F   579 ;       failure, check the halt code, and start at location 100. If the RPB is
                                     020F   580 ;       intrepreted properly a halt code of 12 will be printed on the console
                                     020F   581 ;       terminal. At this point the machine will be in console mode and the
                                     020F   582 ;       operator should type 'S 100" followed by a carrage return to continue
                                     020F   583 ;       testing. If the expected halt code is not printed on the console
                                     020F   584 ;       terminal then an error has occured and the warm start flag was not
                                     020F   585 ;       recognized.
                                     020F   586 ;
                                     020F   587 ; SUBTEST STEPS:
                                     020F   588 ;
                                     020F   589 ;       SUBTEST WARM START FLAG
                                     020F   590 ;       :  Setup RPB at location 0 with warm start flag set
                                     020F   591 ;       :  Send instructions to operator
                                     020F   592 ;       :  Recover from fatal power fail
                                     020F   593 ;       ENDSUBTEST WARM START FLAG
                                     020F   594 ;
                                     020F   595 ; ERRORS:
                                     020F   596 ;
                                     020F   597 ;
                                     020F   598 ; DEBUG:
                                     020F   599 ;
                                     020F   600 ;
                                     020F   601 ;-
                                     020F   602         $DS_BGNSUB
                                     020F       T2_S4::
     00010030 9F    00000024'EF   FA  020F                       CALLG   $S$, @#DS$BGNSUB
                    00000B3B'EF   16  021A   603         JSB     CLEAR_BUFFER    ; clear buffer area
                    00000019'EF   94  0220   604         CLRB    B_FLAGS         ; initialize flags
                    0000001A'EF   94  0226   605         CLRB    B_FLAGS2        ; init flags2
                        55   00   D0  022C   606         MOVL    #0,R5           ; load address to build RPB
                        85   00   D0  022F   607         MOVL    #0,(R5)+        ; load first longword with the address
              85    000009CC'EF   DE  0232   608         MOVAL   POWER_UP,(R5)+  ; load address of restart routine
              85    00000008'EF   D0  0239   609         MOVL    L_CHECKSUM,(R5)+ ; load good restart routine checksum
                        65   01   D0  0240   610         MOVL    #1,(R5)         ; clear warm start flag
     00000100'EF    0000001B'EF   7D  0243   611         MOVQ    JUMP,STRTUP     ; load location 100 with jump instruction
                                     024E   612         $DS_PRINTF_S    FORMAT=T_MESSAGE_4
                    0000026D'EF   9F  024E                       PUSHAB  T_MESSAGE_4
              000100F0 9F    01   FB  0254                       CALLS   #$$N, @#DS$PRINTF
                                     025B   613 POWER_DOWN_4:
                                     025B   614         $DS_BREAK
              00010058 9F    6E   FA  025B                       CALLG   (SP), @#DS$BREAK
           02 00000019'EF    01   E0  0262   615         BBS     #V_PWRUP_DONE,B_FLAGS,1$ ; br = power up sequence complete
                             EF   11  026A   616         BRB     POWER_DOWN_4    ; wait for power fail interrupt
                                     026C   617
           1A 00000019'EF    00   E0  026C   618 1$:     BBS     #V_PWRDN_DONE,B_FLAGS,3$ ; br = power down sequence complete
                                     0274   619         $DS_ERRHARD_S   UNIT=#0, MSGADR=T_PWRDN_ERROR
```

```
                            00   DD  0274                           PUSHL   #0
                   000007D3'EF   DF  0276                           PUSHAL  T_PWRDN_ERROR
                            00   DD  027C                           PUSHL   #0
                            01   DD  027E                           PUSHL   #$ER
                                     0280       ;;; TEST 2, SUBTEST 4, ERROR 1
              000100D0 9F   04   FB  0280                           CALLS   #$$M, @#DS$ERRHARD
                                     0287   620         $DS_ABORT
              00010020 9F   6C   FA  0287                           CALLG   (AP), @#DS$ABORT
                                     028E   621 3$:
                                     028E   622         $DS_ENDSUB
                                     028E       T2_S4_X:
     00010038 9F   00000024'EF   FA  028E                           CALLG   $S$, @#DS$ENDSUB
                                     0299   623         $DS_CLRVEC_S    VECTOR=#SCB$L_POWER
                            0C   DD  0299                           PUSHL   #SCB$L_POWER
              00010168 9F   01   FB  029B                           CALLS   #1, @#DS$CLRVEC
                                     02A2   624         $DS_ENDTEST
                       50   01   D0  02A2                           MOVL    #1, R0          ; NORMAL EXIT
                                     02A5       TEST_002_X::
              00010058 9F   6E   FA  02A5                           CALLG   (SP), @#DS$BREAK
                                 04  02AC                           RET                     ; RETURN TO TEST SEQUENCER
                                     02AD   625
                                     02AD   626         $DS_PAGE
                                 02AD                 .SBTTL  TEST 3:   POWER FAIL (BREAKER)
                             00000000                         .PSECT  TEST_003, PAGE, NOWRT
                                 001C
                                 001C    629 ;++
                                 001C    630 ; Test description:
                                 001C    631 ;
                                 001C    632 ;       This test consists of a series of manual operator intervention
                                 001C    633 ;       subtests. These subtests are the same as in the last test but instead
                                 001C    634 ;       of pushing the init switch to simulate the power failure, the breaker
                                 001C    635 ;       on the back panel of the 11-750 is thrown causing a power failure.
                                 001C    636 ;       The operator is asked if the system under test is equipped with a
                                 001C    637 ;       battery backup. If a negative responce is received the test is exited
                                 001C    638 ;       and the power fail test is complete.
                                 001C    639 ;
                                 001C    640 ;
                                 001C    641 ; Assumptions:
                                 001C    642 ;
                                 001C    643 ;       For this test to function properly the right front panel select
                                 001C    644 ;       switch must be in the HALT/RESTART position and the key switch
                                 001C    645 ;       must be in the LOCAL position.
                                 001C    646 ;
                                 001C    647 ;--
                                 001C    648
                                 001C    649         $DS_BGNTEST     <MANUAL,POWER_FAIL,ALL>
                                 001C        DATA_003:
                     00000000    001C                         .LONG   0                  ; TEST ARGUMENT TABLE TERMINATOR
                                 0020        TEST_003::
                         0000    0020                         .WORD   ^M<>               ; ENTRY MASK
                                 0022    650
                                 0022    651         $DS_SETVEC_S    VECTOR=#SCB$L_POWER, SRVADR=POWER_DOWN, CODE=#1
                   01     DD     0022                         PUSHL   #1
           00000924'EF    DF     0024                         PUSHAL  POWER_DOWN
                   0C     DD     002A                         PUSHL   #SCB$L_POWER
      00010160 9F  03     FB     002C                         CALLS   #3, @#DS$SETVEC
                                 0033    652 2$:     $DS_ASKLGCL_S   MSGADR=T_BAT_BKUP, DATADR=B_RESPONCE, YEXFER=3$
                   00     DD     0033                         PUSHL   #0
                   00     DD     0035                         PUSHL   #PAR$_NO
                   00     DD     0037                         PUSHL   #0
           0000005A'EF    DF     0039                         PUSHAL  3$
                   00     DD     003F                         PUSHL   #0
           00000018'EF    DF     0041                         PUSHAL  B_RESPONCE
           00000692'EF    DF     0047                         PUSHAL  T_BAT_BKUP
      00010098 9F  07     FB     004D                         CALLS   #7, @#DS$ASKLGCL
                                 0054    653         $DS_EXIT        TEST
               50  01     D0     0054                         MOVL    #1,R0              ; NOT AN ERROR
                 025F     31     0057                         BRW     TEST_003_X         ; EXIT TEST 3
           00000B20'EF    16     005A    654 3$:     JSB      CALC_CHECKSUM   ; calculate good checksum for RPB.
                                 0060    655         $DS_PRINTF_S    FORMAT=T_INST
           00000021'EF    9F     0060                         PUSHAB  T_INST
      000100F0 9F  01     FB     0066                         CALLS   #$$N, @#DS$PRINTF
                                 006D    656         $DS_PAGE
```

```
                                     006D               .SBTTL  SUBTEST 3.1:  GOOD RPB SUBTEST
                                     006D       ;+
                                     006D       ; SUBTEST DESCRIPTION:
                                     006D       ;
                                     006D       ;       This subtest sets up a good RPB at location zero, which points to
                                     006D       ;       a power up routine that restores all pertinent processor registers,
                                     006D       ;       the argument pointer, frame pointer and stack pointer. The operator
                                     006D       ;       is instructed through the console terminal to throw the circuit
                                     006D       ;       breaker on the rear panel to cause a CPU power failure. If the
                                     006D       ;       RPB is intrepreted properly the instrucions for the next subtest will
                                     006D       ;       be printed on the console terminal. If an error occured, either the
                                     006D       ;       processor will halt unexpectedly or an error report will be generated
                                     006D       ;       by the diagnostic supervisor.
                                     006D       ;
                                     006D       ;
                                     006D       ; SUBTEST STEPS:
                                     006D       ;
                                     006D       ;       SUBTEST GOOD RPB
                                     006D       ;       :  Setup good RPB at location 0
                                     006D       ;       :  Send instructions to operator
                                     006D       ;       :  Recover from power fail
                                     006D       ;       ENDSUBTEST GOOD RPB
                                     006D       ;
                                     006D       ;
                                     006D       ; ERRORS:
                                     006D       ;
                                     006D       ;
                                     006D       ; DEBUG:
                                     006D       ;
                                     006D       ;
                                     006D       ;-
                                     006D               $DS_BGNSUB
                                     006D       T3_S1::
     00010030 9F   00000030'EF   FA  006D                       CALLG   $S$, @#DS$BGNSUB
                                     0078
                   00000B3B'EF   16  0078               JSB     CLEAR_BUFFER    ; clear buffer area
                   00000019'EF   94  007E               CLRB    B_FLAGS         ; initialize flags
                   0000001A'EF   94  0084               CLRB    B_FLAGS2        ; init flags2
                        55  00   D0  008A               MOVL    #0,R5           ; load address to build RPB
                        85  00   D0  008D               MOVL    #0,(R5)+        ; load first longword with the address
              85   000009CC'EF   DE  0090               MOVAL   POWER_UP,(R5)+  ; load address of restart routine
              85   00000008'EF   D0  0097               MOVL    L_CHECKSUM,(R5)+ ; load good restart routine checksum
                            65   D4  009E               CLRL    (R5)            ; clear warm start flag
                                     00A0               $DS_PRINTF_S    FORMAT=T_MESSAGE_5
                   00000352'EF   9F  00A0                       PUSHAB  T_MESSAGE_5
              000100F0 9F   01   FB  00A6                       CALLS   #$$N, @#DS$PRINTF
                                     00AD
                                     00AD       POWER_DOWN_5:
                                     00AD               $DS_BREAK
              00010058 9F   6E   FA  00AD                       CALLG   (SP), @#DS$BREAK
           02 00000019'EF   01   E0  00B4               BBS     #V_PWRUP_DONE,B_FLAGS,1$ ; br = power up sequence complete
                            EF   11  00BC               BRB     POWER_DOWN_5    ; wait for power fail interrupt
                                     00BE
           1A 00000019'EF   00   E0  00BE       1$:     BBS     #V_PWRDN_DONE,B_FLAGS,3$ ; br = power down sequence complete
                                     00C6               $DS_ERRHARD_S   UNIT=#0, MSGADR=T_PWRDN_ERROR
                            00   DD  00C6                           PUSHL   #0
                   000007D3'EF   DF  00C8                           PUSHAL  T_PWRDN_ERROR
                            00   DD  00CE                           PUSHL   #0
                            01   DD  00D0                           PUSHL   #$ER
                                     00D2        ;;; TEST 3, SUBTEST 1, ERROR 1
              000100D0 9F   04   FB  00D2                           CALLS   #$$M, @#DS$ERRHARD
                                     00D9   708          $DS_ABORT
              00010020 9F   6C   FA  00D9                           CALLG   (AP), @#DS$ABORT
                                     00E0   709 3$:
                                     00E0   710
                                     00E0   711          $DS_ENDSUB
                                     00E0        T3_S1_X:
     00010038 9F   00000030'EF   FA  00E0                           CALLG   $S$, @#DS$ENDSUB
                                     00EB   712
                                     00EB   713          $DS_PAGE
                                     00EB               .SBTTL  SUBTEST 3.2:  BAD CHECKSUM SUBTEST
                                     00EB   716 ;+
                                     00EB   717 ; SUBTEST DESCRIPTION:
                                     00EB   718 ;
                                     00EB   719 ;       This subtest sets up an RPB at location zero with a bad checksum.
                                     00EB   720 ;       It points to a power up routine that restores all pertinent processor
                                     00EB   721 ;       registers, the argument pointer, frame pointer and stack pointer.
                                     00EB   722 ;       The operator is instructed through the console terminal to throw the
                                     00EB   723 ;       circuit breaker the rear panel to cause a CPU power failure,
                                     00EB   724 ;       check the halt code, and start a the indicated location. If the RPB is
                                     00EB   725 ;       intrepreted properly a halt code of 11 will be printed on the console
                                     00EB   726 ;       terminal. At this point the machine will be in console mode and the
                                     00EB   727 ;       operator should type "S 100' followed by a carrage return to continue
                                     00EB   728 ;       testing. If the expected halt code is not printed on the console
                                     00EB   729 ;       terminal then an error has occured.
                                     00EB   730 ;
                                     00EB   731 ;
                                     00EB   732 ; SUBTEST STEPS:
                                     00EB   733 ;
                                     00EB   734 ;       SUBTEST BAD CHECKSUM
                                     00EB   735 ;       :  Setup RPB at location 0 with bad checksum
                                     00EB   736 ;       :  Send instructions to operator
                                     00EB   737 ;       :  Recover from fatal power fail
                                     00EB   738 ;       ENDSUBTEST BAD CHECKSUM
                                     00EB   739 ;
                                     00EB   740 ;
                                     00EB   741 ; ERRORS:
                                     00EB   742 ;
                                     00EB   743 ;
                                     00EB   744 ; DEBUG:
                                     00EB   745 ;
                                     00EB   746 ;
                                     00EB   747 ;-
                                     00EB   748         $DS_BGNSUB
                                     00EB       T3_S2::
     00010030 9F    0000003C'EF  FA  00EB                       CALLG   $S$, @#DS$BGNSUB
                    00000B3B'EF  16  00F6   749         JSB     CLEAR_BUFFER    ; clear buffer area
                    00000019'EF  94  00FC   750         CLRB    B_FLAGS         ; initialize flags
                    0000001A'EF  94  0102   751         CLRB    B_FLAGS2        ; init flags2
                       55    00  D0  0108   752         MOVL    #0,R5           ; load address to build RPB
                       85    00  D0  010B   753         MOVL    #0,(R5)+        ; load first longword with the address
              85    000009CC'EF  DE  010E   754         MOVAL   POWER_UP,(R5)+  ; load address of restart routine
              65    00000008'EF  D0  0115   755         MOVL    L_CHECKSUM,(R5) ; load good restart routine checksum
                       85    01  C2  011C   756         SUBL2   #1,(R5)+        ; make the checksum bad
                             65  D4  011F   757         CLRL    (R5)            ; clear warm start flag
     00000100'EF    0000001B'EF  7D  0121   758         MOVQ    JUMP,STRTUP     ; load addr 100 with JMP to restart test
                                     012C   759         $DS_PRINTF_S    FORMAT=T_MESSAGE_6
                    000003EC'EF  9F  012C                       PUSHAB  T_MESSAGE_6
              000100F0 9F    01  FB  0132                       CALLS   #$$N, @#DS$PRINTF
                                     0139   760
                                     0139   761 POWER_DOWN_6:
                                     0139   762         $DS_BREAK
              00010058 9F    6E  FA  0139                       CALLG   (SP), @#DS$BREAK
           02 00000019'EF    01  E0  0140   763         BBS     #V_PWRUP_DONE,B_FLAGS,1$ ; br = power up sequence complete
                             EF  11  0148   764         BRB     POWER_DOWN_6    ; wait for power fail interrupt
```

```
                                      014A    765
          1A 00000019'EF    00    E0  014A    766 1$:     BBS     #V_PWRDN_DONE,B_FLAGS,3$ ; br = power down sequence complete
                                      0152    767         $DS_ERRHARD_S   UNIT=#0, MSGADR=T_PWRDN_ERROR
                            00    DD  0152                        PUSHL   #0
                000007D3'EF       DF  0154                        PUSHAL  T_PWRDN_ERROR
                            00    DD  015A                        PUSHL   #0
                            01    DD  015C                        PUSHL   #$ER
                                      015E            ;;; TEST 3, SUBTEST 2, ERROR 1
             000100D0 9F    04    FB  015E                        CALLS   #$$M, @#DS$ERRHARD
                                      0165    768         $DS_ABORT
             00010020 9F    6C    FA  0165                        CALLG   (AP), @#DS$ABORT
                                      016C    769 3$:
                                      016C    770         $DS_ENDSUB
                                      016C            T3_S2_X:
     00010038 9F    0000003C'EF   FA  016C                        CALLG   $S$, @#DS$ENDSUB
                                      0177    771
                                      0177    772         $DS_PAGE
                                      0177    773
```

```
                                     0177               .SBTTL  SUBTEST 3.3:  SEARCH FOR GOOD RPB
                                     0177   776 ;+
                                     0177   777 ; SUBTEST DESCRIPTION:
                                     0177   778 ;
                                     0177   779 ;       This subtest sets up an RPB at location zero with a bad checksum.
                                     0177   780 ;       It also sets up a good RPB at a higher page boundary which points
                                     0177   781 ;       to a power up routine that restores all pertinent processor
                                     0177   782 ;       registers, the argument pointer, frame pointer and stack pointer.
                                     0177   783 ;       The operator is instructed through the console terminal to throw the
                                     0177   784 ;       circuit breaker on the rear panel to cause a CPU power failure.
                                     0177   785 ;       If the RPB is intrepreted properly the next subtest instructions will
                                     0177   786 ;       be printed on the console terminal. If an error occured, either the
                                     0177   787 ;       processor will halt unexpectedly or an error report will be generated
                                     0177   788 ;       by the diagnostic supervisor.
                                     0177   789 ;
                                     0177   790 ;
                                     0177   791 ; SUBTEST STEPS:
                                     0177   792 ;
                                     0177   793 ;       SUBTEST SEARCH FOR GOOD RPB
                                     0177   794 ;       :  Setup RPB at location 0 with bad checksum
                                     0177   795 ;       :  Setup good RPB at higher page boundary
                                     0177   796 ;       :  Send instructions to operator
                                     0177   797 ;       ENDSUBTEST SEARCH FOR GOOD RPB
                                     0177   798 ;
                                     0177   799 ;
                                     0177   800 ; ERRORS:
                                     0177   801 ;
                                     0177   802 ;
                                     0177   803 ; DEBUG:
                                     0177   804 ;
                                     0177   805 ;
                                     0177   806 ;-
                                     0177   807         $DS_BGNSUB
                                     0177       T3_S3::
     00010030 9F   00000048'EF    FA  0177                   CALLG   $S$, @#DS$BGNSUB
                   00000B3B'EF    16  0182   808        JSB     CLEAR_BUFFER    ; clear buffer area
                   00000019'EF    94  0188   809        CLRB    B_FLAGS         ; initialize flags
                   0000001A'EF    94  018E   810        CLRB    B_FLAGS2        ; init flags2
                       55    00   D0  0194   811        MOVL    #0,R5           ; load address to build bad RPB
                       85    00   D0  0197   812        MOVL    #0,(R5)+        ; load first longword with the address
              85   000009CC'EF    DE  019A   813        MOVAL   POWER_UP,(R5)+  ; load address of restart routine
              65   00000008'EF    D0  01A1   814        MOVL    L_CHECKSUM,(R5) ; load good restart routine checksum
                       85    01   C0  01A8   815        ADDL2   #1,(R5)+        ; make checksum bad
                             65   D4  01AB   816        CLRL    (R5)            ; clear warm start flag
              55   00000000'FF    DE  01AD   817        MOVAL   @A_BUFFER_BEGIN,R5 ; load address to build good RPB
              85   00000000'FF    DE  01B4   818        MOVAL   @A_BUFFER_BEGIN,(R5)+ ; load first longword with the address
              85   000009CC'EF    DE  01BB   819        MOVAL   POWER_UP,(R5)+  ; load address of restart routine
              85   00000008'EF    D0  01C2   820        MOVL    L_CHECKSUM,(R5)+ ; load good restart routine checksum
                             65   D4  01C9   821        CLRL    (R5)            ; clear warm start flag
                                      01CB   822        $DS_PRINTF_S    FORMAT=T_MESSAGE_7
                   000004D0'EF    9F  01CB                      PUSHAB  T_MESSAGE_7
              000100F0 9F    01   FB  01D1                      CALLS   #$$N, @#DS$PRINTF
                                      01D8   823
                                      01D8   824 POWER_DOWN_7:
                                      01D8   825        $DS_BREAK
```

```
             00010058 9F    6E    FA  01D8                         CALLG   (SP), a#DS$BREAK
          02 00000019'EF    01    E0  01DF    826         BBS      #V_PWRUP_DONE,B_FLAGS,1$ ; br = power up sequence complete
                            EF    11  01E7    827         BRB      POWER_DOWN_7     ; wait for power fail interrupt
                                      01E9    828
          1A 00000019'EF    00    E0  01E9    829 1$:     BBS      #V_PWRDN_DONE,B_FLAGS,3$ ; br = power down sequence complete
                                      01F1    830         $DS_ERRHARD_S    UNIT=#0, MSGADR=T_PWRDN_ERROR
                            00    DD  01F1                         PUSHL   #0
                   000007D3'EF    DF  01F3                         PUSHAL  T_PWRDN_ERROR
                            00    DD  01F9                         PUSHL   #0
                            01    DD  01FB                         PUSHL   #$ER
                                      01FD        ;;; TEST 3, SUBTEST 3, ERROR 1
             000100D0 9F    04    FB  01FD                         CALLS   #$$M, a#DS$ERRHARD
                                      0204    831         $DS_ABORT
             00010020 9F    6C    FA  0204                         CALLG   (AP), a#DS$ABORT
                                      020B    832 3$:
                                      020B    833         $DS_ENDSUB
                                      020B        T3_S3_X:
    00010038 9F     00000048'EF    FA  020B                         CALLG   $S$, a#DS$ENDSUB
                                      0216    834
                                      0216    835         $DS_PAGE
                                      0216    836
                                      0216            .SBTTL  SUBTEST 3.4:  WARM START FLAG SUBTEST
                                      0216    839 ;+
                                      0216    840 ; SUBTEST DESCRIPTION:
                                      0216    841 ;
                                      0216    842 ;       This subtest sets up an RPB at location zero with the warm start flag
                                      0216    843 ;       set. The RPB points to a power up routine that restores all pertinent
                                      0216    844 ;       processor registers, the argument pointer, frame pointer and stack
                                      0216    845 ;       pointer. The operator is instructed through the console terminal to
                                      0216    846 ;       throw the circuit breaker on the rear panel to cause a CPU power
                                      0216    847 ;       failure, check the halt code, and start at location 100. If the RPB is
                                      0216    848 ;       interpreted properly a halt code of 12 will be printed on the console
                                      0216    849 ;       terminal. At this point the machine will be in console mode and the
                                      0216    850 ;       operator should type 'S 100' followed by a carrage return to continue
                                      0216    851 ;       testing. If the expected halt code is not printed on the console
                                      0216    852 ;       terminal then an error has occured and the warm start flag was not
                                      0216    853 ;       recognized.
                                      0216    854 ;
                                      0216    855 ; SUBTEST STEPS:
                                      0216    856 ;
                                      0216    857 ;       SUBTEST WARM START FLAG
                                      0216    858 ;       :  Setup RPB at location 0 with warm start flag set
                                      0216    859 ;       :  Send instructions to operator
                                      0216    860 ;       :  Recover from fatal power fail
                                      0216    861 ;       ENDSUBTEST WARM START FLAG
                                      0216    862 ;
                                      0216    863 ; ERRORS:
                                      0216    864 ;
                                      0216    865 ;
                                      0216    866 ; DEBUG:
                                      0216    867 ;
                                      0216    868 ;
                                      0216    869 ;-
                                      0216    870         $DS_BGNSUB
                                      0216        T3_S4::
     00010030 9F    00000054'EF   FA  0216                    CALLG   $S$, a#DS$BGNSUB
                    00000B3B'EF   16  0221    871         JSB     CLEAR_BUFFER     ; clear buffer area
                    00000019'EF   94  0227    872         CLRB    B_FLAGS          ; initialize flags
                    0000001A'EF   94  022D    873         CLRB    B_FLAGS2         ; init flags2
                         55   00  D0  0233    874         MOVL    #0,R5            ; load address to build RPB
                         85   00  D0  0236    875         MOVL    #0,(R5)+         ; load first longword with the address
               85   000009CC'EF   DE  0239    876         MOVAL   POWER_UP,(R5)+   ; load address of restart routine
               85   00000008'EF   D0  0240    877         MOVL    L_CHECKSUM,(R5)+ ; load good restart routine checksum
                         65   01  D0  0247    878         MOVL    #1,(R5)          ; clear warm start flag
     00000100'EF    0000001B'EF   7D  024A    879         MOVQ    JUMP,STRTUP      ; load location 100 with jump instruction
                                      0255    880         $DS_PRINTF_S    FORMAT=T_MESSAGE_8
                    00000568'EF   9F  0255                    PUSHAB  T_MESSAGE_8
          000100F0 9F         01  FB  025B                    CALLS   #$$N, a#DS$PRINTF
                                      0262    881         $DS_PRINTF_S    FORMAT=T_SWITCH
                    00000651'EF   9F  0262                    PUSHAB  T_SWITCH
          000100F0 9F         01  FB  0268                    CALLS   #$$N, a#DS$PRINTF
                                      026F    882 POWER_DOWN_8:
                                      026F    883         $DS_BREAK
          00010058 9F         6E  FA  026F                    CALLG   (SP), a#DS$BREAK
       02 00000019'EF         01  E0  0276    884         BBS     #V_PWRUP_DONE,B_FLAGS,1$ ; br = power up sequence complete
                              EF  11  027E    885         BRB     POWER_DOWN_8     ; wait for power fail interrupt
```

```
                             0280    886
     1A 00000019'EF    00   E0  0280    887 1$:     BBS     #V_PWRDN_DONE,B_FLAGS,3$ ; br = power down sequence complete
                             0288    888            $DS_ERRHARD_S   UNIT=#0, MSGADR=T_PWRDN_ERROR
                       00   DD  0288                        PUSHL   #0
           000007D3'EF  DF  028A                        PUSHAL  T_PWRDN_ERROR
                       00   DD  0290                        PUSHL   #0
                       01   DD  0292                        PUSHL   #$ER
                             0294        ;;; TEST 3, SUBTEST 4, ERROR 1
        000100D0 9F    04   FB  0294                        CALLS   #$$M, @#DS$ERRHARD
                             029B    889            $DS_ABORT
        00010020 9F    6C   FA  029B                        CALLG   (AP), @#DS$ABORT
                             02A2    890 3$:
                             02A2    891
                             02A2    892            $DS_ENDSUB
                             02A2        T3_S4_X:
  00010038 9F    00000054'EF    FA  02A2                        CALLG   $S$, @#DS$ENDSUB
                             02AD    893
                             02AD    894            $DS_CLRVEC_S    VECTOR=#SCB$L_POWER
                       0C   DD  02AD                        PUSHL   #SCB$L_POWER
        00010168 9F    01   FB  02AF                        CALLS   #1, @#DS$CLRVEC
                             02B6    895
                             02B6    896 ;          $DS_ENDTEST     ;ELIMINATED DUE TO EXTRA CODE NEEDED, VERSION 3.4
                 50    01   D0  02B6    897            MOVL    #1,R0                   ; NORMAL EXIT
                             02B9    898 TEST 003_X::
        00010058 9F    6E   FA  02B9    899            CALLG   (SP),@#DS$BREAK
                 19    00   DA  02C0    900            MTPR    #0,#^X19                ; CLEAR THE NICR
        18   80000090 8F    DA  02C3    901            MTPR    #^X80000090,#^X18       ; TRANSFER NICR TO ICR, MAKE SURE
                             02CA    902                                            ; IR AND ERROR ARE CLEAR
                            04  02CA    903            RET                             ; RETURN
                             02CB    904
                             02CB    905            $DS_ENDMOD
                        00000000                            .PSECT  $TSTCNT, NOEXE, NOWRT, OVR, LONG
                   00000002  0000                            .LONG   $TN
                             0004    906
                             0004    907            .END
```

```
$$$                             = 00000054 R     05        DS$HELP                           000101B0
$$E                             = 00000001                 DS$INITSCB                        00010170
$$M                             = 00000004                 DS$INLOOP                         00010048
$$N                             = 00000000                 DS$K_ERROR                      = 00000002
$$S                             = 00000004                 DS$K_NORMAL                     = 00000001
$ENV                            = 00000001  G              DS$K_SEVERE                     = 00000004
$ER                             = 00000001                 DS$K_SUBSYS                     = 00000066
$MO                             = 00000001  G              DS$K_WARNING                    = 00000000
$S$                             = 00000054 R     05        DS$LOAD                           00010198
$ST                             = FFFFFFFF                 DS$LOADPCS                        000101C0
$TN                             = 00000002                 DS$MAPDBGBLOCK                    00010118
ALL                             = 00000009  G              DS$MEMSIZE                        000101D8
A_BUFFER_BEGIN                    ********   X   01        DS$MMOFF                          00010158
A_PWR_BUFFER                      ********   X   01        DS$MMON                           00010150
BASE_002                          00000000 R     03        DS$MOVPHY                         00010148
BASE_003                          00000000 R     06        DS$MOVVRT                         00010140
B_FLAGS                           00000019 R     01        DS$PARSE                          000100B8
B_FLAGS2                          0000001A R     01        DS$PRINTB                         000100E0
B_RESPONCE                        00000018 R     01        DS$PRINTF                         000100F0
CALC_CHECKSUM                     00000B20 R     01        DS$PRINTS                         000100F8
CEP_FUNCTIONAL                  = 00000000                 DS$PRINTSIG                       00010100
CEP_REPAIR                      = 00000001                 DS$PRINTX                         000100E8
CLEAR_BUFFER                      00000B3B R     01        DS$PROBE                          000101A0
DATA_002                          00000020 R     03        DS$RELBUF                         00010128
DATA_003                          0000001C R     06        DS$RELMEM                         00010138
DEFAULT                         = 00000000  G              DS$SERVICE_DISPATCHER             000101D0
DS$ABORT                          00010020                 DS$SETIPL                         00010178
DS$ASKADR                         00010090                 DS$SETMAP                         00010188
DS$ASKDATA                        00010080                 DS$SETPRIEXV                      00010110
DS$ASKLGCL                        00010098                 DS$SETVEC                         00010160
DS$ASKSTR                         000100A0                 DS$SHOCHAN                        00010190
DS$ASKVLD                         00010088                 DS$SUMMARY                        00010028
DS$ATTACH                         000101A8                 DS$WAITMS                         00010060
DS$BGNSUB                         00010030                 DS$WAITUS                         00010068
DS$BRANCH                         000100A8                 DS$_ARITH                       = 006600D0
DS$BREAK                          00010058                 DS$_ASBE                        = 00660118
DS$CANWAIT                        00010070                 DS$_BADLINK                     = 006600F0
DS$CHANNEL                        00010180                 DS$_BADTYPE                     = 006600E8
DS$CKLOOP                         00010040                 DS$_BIIC                        = 00660120
DS$CLRVEC                         00010168                 DS$_CHME                        = 006600A8
DS$CNTRLC                         00010078                 DS$_CHMK                        = 006600E0
DS$CVTREG                         000100B0                 DS$_DEVNAME                     = 00660108
DS$ENDPASS                        00010010                 DS$_ERROR                       = 00660002
DS$ENDSUB                         00010038                 DS$_FHWE                        = 00660068
DS$ERRDEV                         000100C8                 DS$_FRAGBUF                     = 00660080
DS$ERRHARD                        000100D0                 DS$_ICBUSY                      = 006600C8
DS$ERRPREP                        00010108                 DS$_ICERR                       = 006600C0
DS$ERRSOFT                        000100D8                 DS$_IHWE                        = 00660060
DS$ERRSYS                         000100C0                 DS$_ILLCHAR                     = 00660018
DS$ESCAPE                         00010050                 DS$_ILLPAGCNT                   = 00660078
DS$FREEDBGSYM                     000101B8                 DS$_ILLUNIT                     = 00660100
DS$GETBUF                         00010120                 DS$_INSFMEM                     = 00660050
DS$GETMEM                         00010130                 DS$_IPL2HI                      = 006600B8
DS$GETTERM                        000101C8                 DS$_IVADDR                      = 00660040
DS$GPHARD                         00010018                 DS$_IVVECT                      = 00660038
```

```
DS$_KRNLSTK                  = 00660090                JUMP                            0000001B R      01
DS$_LOGIC                    = 00660070                L_BUF_BEGIN                     00000000 R      01
DS$_MCHK                     = 00660088                L_CHECKSUM                      00000008 R      01
DS$_MMOFF                    = 00660058                L_COUNT                         00000004 R      01
DS$_NEEDUNIT                 = 006600F8                L_IS_SP                         00000010 R      01
DS$_NODE                     = 00660128                L_KS_SP                         00000014 R      01
DS$_NOPCS                    = 00660110                L_PSL                           0000000C R      01
DS$_NORMAL                   = 00660001                MANUAL                        = 00000001  G
DS$_NOSUPPORT                = 006600B1                MCHK                          = 00000005  G
DS$_NOTDON                   = 00660030                MSG_002                         00000002 R      03
DS$_NOTIMP                   = 006600B0                MSG_003                         00000002 R      06
DS$_NULLSTR                  = 00660010                M_BADMODE_DN                  = 00000001
DS$_OVERFLOW                 = 00660008                M_BADMODE_UP                  = 00000002
DS$_POWER                    = 00660098                M_BADSTACK_DN                 = 00000004
DS$_PROGERR                  = 00660020                M_BADSTACK_UP                 = 00000008
DS$_SEVERE                   = 00660004                M_BAD_PC                      = 00000004
DS$_TRANSL                   = 006600A0                M_PWRDN_DONE                  = 00000001
DS$_TRUNCATE                 = 00660028                M_PWRUP_DONE                  = 00000002
DS$_UNEXPINT                 = 006600D8                PAR$M_ATDEF                   = 00000010
DS$_VASFULL                  = 00660048                PAR$M_ATHI                    = 00000008
DS$_WARNING                  = 00660000                PAR$M_ATLO                    = 00000004
DSA$AL_APTMAIL                 0000FE00                PAR$M_NODEF                   = 00000002
DSA$AT_APTTXT                  0000FA00                PAR$V_ATDEF                   = 00000004
DSA$GL_APTCOM                  0000FE04                PAR$V_ATHI                    = 00000003
DSA$GL_DEVLEN                  0000FE58                PAR$V_ATLO                    = 00000002
DSA$GL_ERRNO                   0000FE44                PAR$V_NODEF                   = 00000001
DSA$GL_EVENT                   0000FE48                PAR$_BIN                      = 00000002
DSA$GL_FLAGS                   0000FE00                PAR$_DEC                      = 0000000A
DSA$GL_MSGTYP                  0000FE40                PAR$_HEX                      = 00000010
DSA$GL_PASSES                  0000FE08                PAR$_NO                       = 00000000
DSA$GL_PASSNO                  0000FE54                PAR$_OCT                      = 00000008
DSA$GL_SECTNO                  0000FE10                PAR$_YES                      = 00000001
DSA$GL_SID                     0000FE14                PCS                           = 00000008  G
DSA$GL_SUBTNO                  0000FE4C                POWER_DOWN                      00000924 R      01
DSA$GL_TESTNO                  0000FE50                POWER_DOWN_1                    000000A6 R      03
DSA$GL_UNITS                   0000FE0C                POWER_DOWN_2                    00000132 R      03
DSA$GQ_MSGPTR                  0000FE68                POWER_DOWN_3                    000001D1 R      03
DSA$GT_DEVNAM                  0000FE5C                POWER_DOWN_4                    0000025B R      03
DSA$V_OPER                   = 0000000C                POWER_DOWN_5                    000000AD R      06
ENV$M_DOMAIN                 = 00000002                POWER_DOWN_6                    00000139 R      06
ENV$M_LEVEL                  = 00000001                POWER_DOWN_7                    000001D8 R      06
ENV$M_SUPER                  = 000003FC                POWER_DOWN_8                    0000026F R      06
ENV$S_DOMAIN                 = 00000001                POWER_FAIL                    = 00000003  G
ENV$S_LEVEL                  = 00000001                POWER_UP                        000009CC R      01
ENV$S_SUPER                  = 00000008                PR$_ASTLVL                      ********   X   01
ENV$V_DOMAIN                 = 00000001                PR$_ESP                         ********   X   01
ENV$V_LEVEL                  = 00000000                PR$_ISP                         ********   X   01
ENV$V_SUPER                  = 00000002                PR$_KSP                         ********   X   01
ENV$_CPU                     = 00000000                PR$_P0BR                        ********   X   01
ENV$_FUNCTIONAL              = 00000000                PR$_P0LR                        ********   X   01
ENV$_REPAIR                  = 00000001                PR$_P1BR                        ********   X   01
ENV$_SUPER                   = 00000001                PR$_P1LR                        ********   X   01
ENV$_SYSTEM                  = 00000001                PR$_PCBB                        ********   X   01
HALT                         = 00000004   G            PR$_SBR                         ********   X   01
IPR                          = 00000006   G            PR$_SCBB                        ********   X   01
```

```
PR$_SLR                         ********    X    01       SEP_REPAIR                      = 00000003
PR$_SSP                         ********    X    01       SIZ...                          = 00000001
PR$_USP                         ********    X    01       STRTUP                          = 00000100
PSL$K_EXEC                    = 00000001                  SYS$ALLOC                         00010238
PSL$K_KERNEL                  = 00000000                  SYS$ASCTIM                        00010248
PSL$K_SUPER                   = 00000002                  SYS$ASSIGN                        00010250
PSL$K_USER                    = 00000003                  SYS$BINTIM                        00010258
PSL$M_CBIT                    = 00000001                  SYS$CANCEL                        00010260
PSL$M_NBIT                    = 00000008                  SYS$CANTIM                        00010268
PSL$M_VBIT                    = 00000002                  SYS$CLOSE                         000105B8
PSL$M_ZBIT                    = 00000004                  SYS$CLREF                         00010298
PSL$S_CURMOD                  = 00000002                  SYS$CONNECT                       000105C0
PSL$V_CBIT                    = 00000000                  SYS$DALLOC                        000102D8
PSL$V_CURMOD                  = 00000018                  SYS$DASSGN                        000102E0
PSL$V_IS                      = 0000001A                  SYS$DISCONNECT                    000105D0
PSL$V_NBIT                    = 00000003                  SYS$FAO                           00010350
PSL$V_VBIT                    = 00000001                  SYS$FAOL                          00010358
PSL$V_ZBIT                    = 00000002                  SYS$GET                           00010580
SCB$L_ACCESS                    00000020                  SYS$GETCHN                        000104C8
SCB$L_ARITH                     00000034                  SYS$GETTIM                        00010378
SCB$L_BREAK                     0000002C                  SYS$LKWSET                        000103A0
SCB$L_CHME                      00000044                  SYS$NUMTIM                        000103B8
SCB$L_CHMK                      00000040                  SYS$OPEN                          00010608
SCB$L_CHMS                      00000048                  SYS$QIO                           000103C8
SCB$L_CHMU                      0000004C                  SYS$QIOW                          00010200
SCB$L_COMPAT                    00000030                  SYS$READ                          00010590
SCB$L_KNLSTK                    00000008                  SYS$READEF                        000103D0
SCB$L_MACHCHK                   00000004                  SYS$SETAST                        000103F8
SCB$L_OPCCUS                    00000014                  SYS$SETEF                         00010400
SCB$L_OPCDEC                    00000010                  SYS$SETIMR                        00010420
SCB$L_POWER                     0000000C                  SYS$SETPRI                        00010428
SCB$L_RADRMOD                   0000001C                  SYS$SETPRT                        00010430
SCB$L_ROPRAND                   00000018                  SYS$SETRWM                        00010438
SCB$L_RXDB                      000000F8                  SYS$SETSWM                        00010448
SCB$L_SFTLVL1                   00000084                  SYS$ULKPAG                        00010460
SCB$L_SFTLVL10                  000000A8                  SYS$ULWSET                        00010468
SCB$L_SFTLVL11                  000000AC                  SYS$UNWIND                        00010470
SCB$L_SFTLVL12                  000000B0                  SYS$WAITFR                        00010478
SCB$L_SFTLVL13                  000000B4                  SYS$WFLAND                        00010488
SCB$L_SFTLVL14                  000000B8                  SYS$WFLOR                         00010490
SCB$L_SFTLVL15                  000000BC                  T2_S1                             00000066 RG      03
SCB$L_SFTLVL2                   00000088                  T2_S1_X                           000000D9 R       03
SCB$L_SFTLVL3                   0000008C                  T2_S2                             000000E4 RG      03
SCB$L_SFTLVL4                   00000090                  T2_S2_X                           00000165 R       03
SCB$L_SFTLVL5                   00000094                  T2_S3                             00000170 RG      03
SCB$L_SFTLVL6                   00000098                  T2_S3_X                           00000204 R       03
SCB$L_SFTLVL7                   0000009C                  T2_S4                             0000020F RG      03
SCB$L_SFTLVL8                   000000A0                  T2_S4_X                           0000028E R       03
SCB$L_SFTLVL9                   000000A4                  T3_S1                             0000006D RG      06
SCB$L_TBIT                      00000028                  T3_S1_X                           000000E0 R       06
SCB$L_TIMER                     000000C0                  T3_S2                             000000EB RG      06
SCB$L_TRANSL                    00000024                  T3_S2_X                           0000016C R       06
SCB$L_TXDB                      000000FC                  T3_S3                             00000177 RG      06
SCB$L_ZERO                      00000000                  T3_S3_X                           0000020B R       06
SEP_FUNCTIONAL                = 00000002                  T3_S4                             00000216 RG      06
```

```
T3_S4_X                          000002A2 R      06
TEST_002                         00000024 RG     03
TEST_002_X                       000002A5 RG     03
TEST_003                         00000020 RG     06
TEST_003_X                       000002B9 RG     06
TU58                           = 00000002  G
T_BADMODE                        0000081B R      01
T_BADMODE_DN                     00000843 R      01
T_BADMODE_UP                     0000086D R      01
T_BADSTACK                       00000895 R      01
T_BADSTACK_DN                    000008C4 R      01
T_BADSTACK_UP                    000008F5 R      01
T_BAT_BKUP                       00000692 R      01
T_INST                           00000021 R      01
T_MESSAGE_1                      00000063 R      01
T_MESSAGE_2                      000000F9 R      01
T_MESSAGE_3                      000001D9 R      01
T_MESSAGE_4                      0000026D R      01
T_MESSAGE_5                      00000352 R      01
T_MESSAGE_6                      000003EC R      01
T_MESSAGE_7                      000004D0 R      01
T_MESSAGE_8                      00000568 R      01
T_NO_OPER                        0000073A R      01
T_PWRDN_ERROR                    000007D3 R      01
T_PWRUP_ERROR                    000007F8 R      01
T_PWRUP_FAILURE                  00000704 R      01
T_SWITCH                         00000651 R      01
V_BADMODE_DN                   = 00000000
V_BADMODE_UP                   = 00000001
V_BADSTACK_DN                  = 00000002
V_BADSTACK_UP                  = 00000003
V_BAD_PC                       = 00000002
V_PWRDN_DONE                   = 00000000
V_PWRUP_DONE                   = 00000001
WCS                            = 00000007  G

                                                +----------------+
                                                ! Psect synopsis !
                                                +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
ECKAXC                          00000B53  ( 2899.)  01 (  1.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD  NOWRT NOVEC PAGE
$ABS$                           00010610  (67088.)  02 (  2.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
TEST_002                        000002AD  (  685.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD  NOWRT NOVEC PAGE
DISPATCH                        00000030  (   48.)  04 (  4.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD  NOWRT NOVEC LONG
ARGLIST                         00000060  (   96.)  05 (  5.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD  NOWRT NOVEC LONG
TEST_003                        000002CB  (  715.)  06 (  6.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD  NOWRT NOVEC PAGE
$TSTCNT                         00000004  (    4.)  07 (  7.)  NOPIC   USR   OVR   REL   LCL NOSHR NOEXE   RD  NOWRT NOVEC LONG
```

```
                                         +-------------------------+
                                         ! Symbol Cross Reference !
                                         +-------------------------+

SYMBOL                          VALUE       DEFINITION      REFERENCES...
------                          -----       ----------      -------------
$$$                            =00000054-R  870     (15)     424    (7)       482    (8)       540    (9)       602    (10)
                                                             689    (12)      748    (13)      807    (14)      870    (15)
$$E                            =00000001    870     (15)     445    (7)       504    (8)       566    (9)       622    (10)
                                                             711    (12)      770    (13)      833    (14)      892    (15)
$$M                            =00000004    888     (15)     315    (5)       318    (5)       320    (5)       324    (5)
                                                             327    (5)       329    (5)       383    (6)       442    (7)
                                                             501    (8)       563    (9)       619    (10)      649    (11)
                                                             707    (12)      767    (13)      830    (14)      888    (15)
$$N                            =00000000    888     (15)     315    (5)       318    (5)       320    (5)       324    (5)
                                                             327    (5)       329    (5)     #-385    (6)     #-390    (6)
                                                           #-434    (7)       442    (7)     #-493    (8)       501    (8)
                                                           #-555    (9)       563    (9)     #-612    (10)      619    (10)
                                                           #-655    (11)    #-699    (12)      707    (12)    #-759    (13)
                                                             767    (13)    #-822    (14)      830    (14)    #-880    (15)
                                                           #-881    (15)      888    (15)      91     (3)
$$S                            =00000004    870     (15)     359    (6)       383    (6)       393    (7)       424    (7)
                                                             449    (8)       482    (8)       508    (9)       540    (9)
                                                             570    (10)      602    (10)      628    (11)      649    (11)
                                                             658    (12)      689    (12)      715    (13)      748    (13)
                                                             775    (14)      807    (14)      838    (15)      870    (15)
$ENV                           =00000001    72      (3)
$ER                            =00000001    892     (15)   #-315    (5)     #-318    (5)     #-320    (5)     #-324    (5)
                                                           #-327    (5)     #-329    (5)       424    (7)     #-442    (7)
                                                             482    (8)     #-501    (8)       540    (9)     #-563    (9)
                                                             602    (10)    #-619    (10)      689    (12)    #-707    (12)
                                                             748    (13)    #-767    (13)      807    (14)    #-830    (14)
                                                             870    (15)    #-888    (15)
$MO                            =00000001    905     (15)     905    (15)
$S$                            =00000054-R  870     (15)     424    (7)       445    (7)       482    (8)       504    (8)
                                                             540    (9)       566    (9)       602    (10)      622    (10)
                                                             689    (12)      711    (12)      748    (13)      770    (13)
                                                             807    (14)      833    (14)      870    (15)      892    (15)
$ST                            =FFFFFFFF    892     (15)     315    (5)       318    (5)       320    (5)       324    (5)
                                                             327    (5)       329    (5)       383    (6)       393    (7)
                                                             424    (7)       442    (7)       445    (7)       449    (8)
                                                             482    (8)       501    (8)       504    (8)       508    (9)
                                                             540    (9)       563    (9)       566    (9)       570    (10)
                                                             602    (10)      619    (10)      622    (10)      624    (10)
                                                             649    (11)      658    (12)      689    (12)      707    (12)
                                                             711    (12)      715    (13)      748    (13)      767    (13)
                                                             770    (13)      775    (14)      807    (14)      830    (14)
                                                             833    (14)      838    (15)      870    (15)      888    (15)
                                                             892    (15)
$TN                            =00000002    905     (15)     315    (5)       318    (5)       320    (5)       324    (5)
                                                             327    (5)       329    (5)       359    (6)       383    (6)
                                                             393    (7)       442    (7)       449    (8)       501    (8)
                                                             508    (9)       563    (9)       570    (10)      619    (10)
                                                             624    (10)      628    (11)      649    (11)      653    (11)
```

```
                                                              658    (12)     707    (12)     715    (13)     767    (13)
                                                              775    (14)     830    (14)     838    (15)     888    (15)
                                                              905    (15)
ALL                            =00000009     91     (3)       383    (6)      649    (11)
A_BUFFER_BEGIN                  00000000-XR                   350    (5)      550    (9)      551    (9)      817    (14)
                                                              818    (14)
A_PWR_BUFFER                    00000000-XR                   246    (5)
BASE_002                        00000000-R   359    (6)       383    (6)
BASE_003                        00000000-R   628    (11)      649    (11)
BIT...                         =00000003     88     (3)        72    (3)       85    (3)       87    (3)       88    (3)
B_FLAGS                         00000019-R   106    (3)     #-274    (5)    #-331    (5)    #-427    (7)      438    (7)
                                                              441    (7)    #-484    (8)      497    (8)      500    (8)
                                                            #-542    (9)      559    (9)      562    (9)    #-604    (10)
                                                              615    (10)     618    (10)   #-692    (12)     703    (12)
                                                              706    (12)   #-750    (13)     763    (13)     766    (13)
                                                            #-809    (14)     826    (14)     829    (14)   #-872    (15)
                                                              884    (15)     887    (15)
B_FLAGS2                        0000001A-R   107    (3)     #-251    (5)    #-253    (5)    #-292    (5)    #-294    (5)
                                                              313    (5)      314    (5)      317    (5)      322    (5)
                                                              323    (5)      326    (5)    #-428    (7)    #-485    (8)
                                                            #-543    (9)    #-605    (10)   #-693    (12)   #-751    (13)
                                                            #-810    (14)   #-873    (15)
B_RESPONCE                      00000018-R   105    (3)       652    (11)
CALC_CHECKSUM                   00000B20-R   338    (5)       389    (6)      654    (11)
CEP_FUNCTIONAL                 =00000000     72     (3)
CEP_REPAIR                     =00000001     72     (3)        72    (3)
CLEAR_BUFFER                    00000B3B-R   349    (5)       426    (7)      483    (8)      541    (9)      603    (10)
                                                              691    (12)     749    (13)     808    (14)     871    (15)
DATA_002                        00000020-R   383    (6)       383    (6)
DATA_003                        0000001C-R   649    (11)      649    (11)
DEFAULT                        =00000000     91     (3)
DS$ABORT                        00010020                      316    (5)      319    (5)      321    (5)      325    (5)
                                                              328    (5)      330    (5)      386    (6)      443    (7)
                                                              502    (8)      564    (9)      620    (10)     708    (12)
                                                              768    (13)     831    (14)     889    (15)
DS$ASKLGCL                      00010098                      652    (11)
DS$BGNSUB                       00010030                      424    (7)      482    (8)      540    (9)      602    (10)
                                                              689    (12)     748    (13)     807    (14)     870    (15)
DS$BREAK                        00010058                      437    (7)      496    (8)      558    (9)      614    (10)
                                                              624    (10)     702    (12)     762    (13)     825    (14)
                                                              883    (15)     899    (15)
DS$CLRVEC                       00010168                      623    (10)     894    (15)
DS$ENDSUB                       00010038                      445    (7)      504    (8)      566    (9)      622    (10)
                                                              711    (12)     770    (13)     833    (14)     892    (15)
DS$ERRHARD                      000100D0                      315    (5)      318    (5)      320    (5)      324    (5)
                                                              327    (5)      329    (5)      442    (7)      501    (8)
                                                              563    (9)      619    (10)     707    (12)     767    (13)
                                                              830    (14)     888    (15)
DS$K_ERROR                     =00000002     85     (3)
DS$K_NORMAL                    =00000001     85     (3)
DS$K_SEVERE                    =00000004     85     (3)
DS$K_SUBSYS                    =00000066     85     (3)        85    (3)
DS$K_WARNING                   =00000000     85     (3)
DS$PRINTF                       000100F0                      385    (6)      390    (6)      434    (7)      493    (8)
                                                              555    (9)      612    (10)     655    (11)     699    (12)
```

```
                                                                 759     (13)      822     (14)      880     (15)      881     (15)
DS$SETVEC                       00010160                         387     (6)       651     (11)
DS$_ARITH                      =006600D0     85       (3)
DS$_ASBE                       =00660118     85       (3)
DS$_BADLINK                    =006600F0     85       (3)
DS$_BADTYPE                    =006600E8     85       (3)
DS$_BIIC                       =00660120     85       (3)
DS$_CHME                       =006600A8     85       (3)
DS$_CHMK                       =006600E0     85       (3)
DS$_DEVNAME                    =00660108     85       (3)
DS$_ERROR                      =00660002     85       (3)
DS$ F' E                       =00660068     85       (3)
DS$_FRAGBUF                    =00660080     85       (3)
DS$ ICBUSY                     =006600C8     85       (3)
DS$_ICERR                      =006600C0     85       (3)
DS$ IHWE                       =00660060     85       (3)
DS$_ILLCHAR                    =00660018     85       (3)
DS$_ILLPAGCNT                  =00660078     85       (3)
DS$_ILLUNIT                    =00660100     85       (3)
DS$_INSFMEM                    =00660050     85       (3)
DS$_IPL2HI                     =006600B8     85       (3)
DS$_IVADDR                     =00660040     85       (3)
DS$_IVVECT                     =00660038     85       (3)
DS$ KRNLSTK                    =00660090     85       (3)
DS$_LOGIC                      =00660070     85       (3)
DS$_MCHK                       =00660088     85       (3)
DS$ MMOFF                      =00660058     85       (3)
DS$_NEEDUNIT                   =006600F8     85       (3)
DS$_NODE                       =00660128     85       (3)
DS$_NOPCS                      =00660110     85       (3)
DS$_NORMAL                     =00660001     85       (3)
DS$ NOSUPPORT                  =006600B1     85       (3)
DS$ NOTDON                     =00660030     85       (3)
DS$_NOTIMP                     =006600B0     85       (3)        85     (3)
DS$_NULLSTR                    =00660010     85       (3)
DS$_OVERFLOW                   =00660008     85       (3)
DS$_POWER                      =00660098     85       (3)
DS$_PROGERR                    =00660020     85       (3)
DS$_SEVERE                     =00660004     85       (3)
DS$_TRANSL                     =006600A0     85       (3)
DS$_TRUNCATE                   =00660028     85       (3)
DS$_UNEXPINT                   =006600D8     85       (3)
DS$_VASFULL                    =00660048     85       (3)
DS$_WARNING                    =00660000     85       (3)        85     (3)
DSA$GL_FLAGS                    0000FE00                         384     (6)
DSA$V_OPER                     =0000000C                       #-384     (6)
ENV$M_DOMAIN                   =00000002     72       (3)
ENV$M_LEVEL                    =00000001     72       (3)
ENV$M_SUPER                    =000003FC     72       (3)
ENV$S_DOMAIN                   =00000001     72       (3)
ENV$S_LEVEL                    =00000001     72       (3)
ENV$S_SUPER                    =00000008     72       (3)
ENV$V_DOMAIN                   =00000001     72       (3)        72     (3)
ENV$V_LEVEL                    =00000000     72       (3)        72     (3)
ENV$V_SUPER                    =00000002     72       (3)
```

```
ENV$_CPU                       =00000000     72       (3)       72      (3)
ENV$_FUNCTIONAL                =00000000     72       (3)       72      (3)
ENV$_REPAIR                    =00000001     72       (3)       72      (3)
ENV$_SUPER                     =00000001     72       (3)
ENV$_SYSTEM                    =00000001     72       (3)       72      (3)
HALT                           =00000004     91       (3)
IPR                            =00000006     91       (3)
JUMP                            0000001B-R   127      (3)     #-492     (8)      #-611     (10)     #-758     (13)     #-879     (15)
L_BUF_BEGIN                     00000000-R   98       (3)     #-273     (5)      #-286     (5)
L_CHECKSUM                      00000008-R   100      (3)     #-344     (5)      #-432     (7)      #-489     (8)      #-547     (9)
                                                              #-553     (9)      #-609     (10)     #-697     (12)     #-755     (13)
                                                              #-814     (14)     #-820     (14)     #-877     (15)
L_COUNT                         00000004-R   99       (3)     #-340     (5)      #-343     (5)      #-351     (5)      #-353     (5)
L_IS_SP                         00000010-R   102      (3)     #-254     (5)      #-257     (5)      #-295     (5)      #-298     (5)
L_KS_SP                         00000014-R   103      (3)     #-255     (5)      #-258     (5)      #-296     (5)      #-299     (5)
L_PSL                           0000000C-R   101      (3)     #-247     (5)        249     (5)        252     (5)      #-288     (5)
                                                                290     (5)        293     (5)
MANUAL                         =00000001     91       (3)       383     (6)        649     (11)
MCHK                           =00000005     91       (3)
MSG_002                         00000002-R   359      (6)       383     (6)
MSG_003                         00000002-R   628      (11)      649     (11)
M_BADMODE_DN                   =00000001     118      (3)     #-251     (5)
M_BADMODE_UP                   =00000002     120      (3)     #-292     (5)
M_BADSTACK_DN                  =00000004     122      (3)     #-253     (5)
M_BADSTACK_UP                  =00000008     124      (3)     #-294     (5)
M_BAD_PC                       =00000004     115      (3)
M_PWRDN_DONE                   =00000001     111      (3)     #-274     (5)
M_PWRUP_DONE                   =00000002     113      (3)     #-331     (5)
PAR$M_ATDEF                    =00000010     88       (3)
PAR$M_ATHI                     =00000008     88       (3)
PAR$M_ATLO                     =00000004     88       (3)
PAR$M_NODEF                    =00000002     88       (3)
PAR$V_ATDEF                    =00000004     88       (3)
PAR$V_ATHI                     =00000003     88       (3)
PAR$V_ATLO                     =00000002     88       (3)
PAR$V_NODEF                    =00000001     88       (3)
PAR$_BIN                       =00000002     88       (3)
PAR$_DEC                       =0000000A     88       (3)
PAR$_HEX                       =00000010     88       (3)
PAR$_NO                        =00000000     88       (3)     #-652     (11)
PAR$_OCT                       =00000008     88       (3)
PAR$_YES                       =00000001     88       (3)
PCS                            =00000008     91       (3)
POWER_DOWN                      00000924-R   245      (5)       387     (6)        651     (11)
POWER_DOWN_1                    000000A6-R   436      (7)     #-439     (7)
POWER_DOWN_2                    00000132-R   495      (8)     #-498     (8)
POWER_DOWN_3                    000001D1-R   557      (9)     #-560     (9)
POWER_DOWN_4                    0000025B-R   613      (10)    #-616     (10)
POWER_DOWN_5                    000000AD-R   701      (12)    #-704     (12)
POWER_DOWN_6                    00000139-R   761      (13)    #-764     (13)
POWER_DOWN_7                    000001D8-R   824      (14)    #-827     (14)
POWER_DOWN_8                    0000026F-R   882      (15)    #-885     (15)
POWER_FAIL                     =00000003     91       (3)       383     (6)        649     (11)
POWER_UP                        000009CC-R   285      (5)       127     (3)        341     (5)        431     (7)        488     (8)
                                                                546     (9)        552     (9)        608     (10)       696     (12)
                                                              754     (13)     813     (14)     819     (14)     876     (15)
PR$_ASTLVL                       00000000-XR                 #-271    (5)     #-300    (5)
PR$_ESP                          00000000-XR                 #-261    (5)     #-310    (5)
PR$_ISP                          00000000-XR                 #-254    (5)     #-295    (5)
PR$_KSP                          00000000-XR                 #-258    (5)     #-299    (5)
PR$_P0BR                         00000000-XR                 #-267    (5)     #-304    (5)
PR$_P0LR                         00000000-XR                 #-268    (5)     #-303    (5)
PR$_P1BR                         00000000-XR                 #-269    (5)     #-302    (5)
PR$_P1LR                         00000000-XR                 #-270    (5)     #-301    (5)
PR$_PCBB                         00000000-XR                 #-266    (5)     #-305    (5)
PR$_SBR                          00000000-XR                 #-264    (5)     #-307    (5)
PR$_SCBB                         00000000-XR                 #-272    (5)     #-287    (5)
PR$_SLR                          00000000-XR                 #-265    (5)     #-306    (5)
PR$_SSP                          00000000-XR                 #-262    (5)     #-309    (5)
PR$_USP                          00000000-XR                 #-263    (5)     #-308    (5)
PSL$K_EXEC                      =00000001      87      (3)
PSL$K_KERNEL                    =00000000      87      (3)   #-249    (5)     #-290    (5)
PSL$K_SUPER                     =00000002      87      (3)
PSL$K_USER                      =00000003      87      (3)
PSL$M_CBIT                      =00000001      87      (3)
PSL$M_NBIT                      =00000008      87      (3)
PSL$M_VBIT                      =00000002      87      (3)
PSL$M_ZBIT                      =00000004      87      (3)
PSL$S_CURMOD                    =00000002                    #-248    (5)     #-289    (5)
PSL$V_CBIT                      =00000000      87      (3)
PSL$V_CURMOD                    =00000018                    #-248    (5)     #-289    (5)
PSL$V_IS                        =0000001A                    #-252    (5)     #-293    (5)
PSL$V_NBIT                      =00000003      87      (3)
PSL$V_VBIT                      =00000001      87      (3)
PSL$V_ZBIT                      =00000002      87      (3)
SCB$L_POWER                      0000000C                    #-387    (6)     #-623    (10)    #-651    (11)    #-894    (15)
SEP_FUNCTIONAL                  =00000002      72      (3)
SEP_REPAIR                      =00000003      72      (3)
SIZ...                          =00000001      88      (3)     72     (3)      87     (3)      88     (3)
STRTUP                          =00000100      129     (3)   #-492    (8)     #-611    (10)    #-758    (13)    #-879    (15)
T2_S1                            00000066-R    424     (7)
T2_S1_X                          000000D9-R    445     (7)
T2_S2                            000000E4-R    482     (8)
T2_S2_X                          00000165-R    504     (8)
T2_S3                            00000170-R    540     (9)
T2_S3_X                          00000204-R    566     (9)
T2_S4                            0000020F-R    602     (10)
T2_S4_X                          0000028E-R    622     (10)
T3_S1                            0000006D-R    689     (12)
T3_S1_X                          000000E0-R    711     (12)
T3_S2                            000000EB-R    748     (13)
T3_S2_X                          0000016C-R    770     (13)
T3_S3                            00000177-R    807     (14)
T3_S3_X                          0000020B-R    833     (14)
T3_S4                            00000216-R    870     (15)
T3_S4_X                          000002A2-R    892     (15)
TEST_002                         00000024-R    383     (6)     383    (6)
TEST_002_X                       000002A5-R    624     (10)
TEST_003                         00000020-R    649     (11)    649    (11)
TEST_003_X                       000002B9-R    898     (15)  #-653    (11)
```

```
TU58                           =00000002     91      (3)
T_BADMODE                       0000081B-R   210      (4)      318     (5)
T_BADMODE_DN                    00000843-R   214      (4)      320     (5)
T_BADMODE_UP                    0000086D-R   218      (4)      315     (5)
T_BADSTACK                      00000895-R   222      (4)      327     (5)
T_BADSTACK_DN                   000008C4-R   226      (4)      329     (5)
T_BADSTACK_UP                   000008F5-R   230      (4)      324     (5)
T_BAT_BKUP                      00000692-R   184      (4)      652     (11)
T_INST                          00000021-R   133      (4)      390     (6)        655     (11)
T_MESSAGE_1                     00000063-R   137      (4)      434     (7)
T_MESSAGE_2                     000000F9-R   142      (4)      493     (8)
T_MESSAGE_3                     000001D9-R   148      (4)      555     (9)
T_MESSAGE_4                     0000026D-R   153      (4)      612     (10)
T_MESSAGE_5                     00000352-R   159      (4)      699     (12)
T_MESSAGE_6                     000003EC-R   164      (4)      759     (13)
T_MESSAGE_7                     000004D0-R   170      (4)      822     (14)
T_MESSAGE_8                     00000568-R   175      (4)      880     (15)
T_NO_OPER                       0000073A-R   196      (4)      385     (6)
T_PWRDN_ERROR                   000007D3-R   202      (4)      442     (7)        501     (8)        563     (9)        619     (10)
                                                               707     (12)       767     (13)       830     (14)       888     (15)
T_PWRUP_ERROR                   000007F8-R   206      (4)
T_PWRUP_FAILURE                 00000704-R   191      (4)
T_SWITCH                        00000651-R   181      (4)      881     (15)
V_BADMODE_DN                   =00000000     117      (3)     #-313    (5)
V_BADMODE_UP                   =00000001     119      (3)     #-314    (5)       #-317    (5)
V_BADSTACK_DN                  =00000002     121      (3)     #-322    (5)
V_BADSTACK_UP                  =00000003     123      (3)     #-323    (5)       #-326    (5)
V_BAD_PC                       =00000002     114      (3)
V_PWRDN_DONE                   =00000000     110      (3)     #-441    (7)       #-500    (8)       #-562    (9)       #-618    (10)
                                                              #-706    (12)      #-766    (13)      #-829    (14)      #-887    (15)
V_PWRUP_DONE                   =00000001     112      (3)     #-438    (7)       #-497    (8)       #-559    (9)       #-615    (10)
                                                              #-703    (12)      #-763    (13)      #-826    (14)      #-884    (15)
WCS                            =00000007     91       (3)
```

```
                                               +-------------------------+
                                               ! Macros Cross Reference !
                                               +-------------------------+

MACRO             SIZE          DEFINITION       REFERENCES...
-----             ----          ----------       -------------
$D1_ABPROGRAM     1             316    (5)       316    (5)       319    (5)       321    (5)       325    (5)       328    (5)
                                                 330    (5)       386    (6)       443    (7)       502    (8)       564    (9)
                                                 620    (10)      708    (12)      768    (13)      831    (14)      889    (15)
$D1_BSUBT         1             424    (7)       424    (7)       482    (8)       540    (9)       602    (10)      689    (12)
                                                 748    (13)      807    (14)      870    (15)
$D1_BTEST         1             383    (6)       383    (6)       649    (11)
$D1_ERR           1             315    (5)       315    (5)       318    (5)       320    (5)       324    (5)       327    (5)
                                                 329    (5)       442    (7)       501    (8)       563    (9)       619    (10)
                                                 707    (12)      767    (13)      830    (14)      888    (15)
$D1_ERR_S         1             315    (5)       315    (5)       318    (5)       320    (5)       324    (5)       327    (5)
                                                 329    (5)       442    (7)       501    (8)       563    (9)       619    (10)
                                                 707    (12)      767    (13)      830    (14)      888    (15)
$D1_ESUBT         1             445    (7)       445    (7)       504    (8)       566    (9)       622    (10)      711    (12)
                                                 770    (13)      833    (14)      892    (15)
$D1_ETEST         1             624    (10)      624    (10)
$D1_EXTEST        1             653    (11)      653    (11)
$D1_PRINT_S       1             385    (6)       385    (6)       390    (6)       434    (7)       493    (8)       555    (9)
                                                 612    (10)      655    (11)      699    (12)      759    (13)      822    (14)
                                                 880    (15)      881    (15)
$D1_SBTTL         2             359    (6)       359    (6)       393    (7)       449    (8)       508    (9)       570    (10)
                                                 628    (11)      658    (12)      715    (13)      775    (14)      838    (15)
$D2_BDATA         1             383    (6)       383    (6)       649    (11)
$D2_BTEST         1             383    (6)       383    (6)       649    (11)
$D2_SBTTL         1             359    (6)       359    (6)       628    (11)
$DEF              1             88     (3)
$DEFINI           1             84     (3)       84     (3)       86     (3)       87     (3)       89     (3)
$DS ABORT         1             316    (5)       316    (5)       319    (5)       321    (5)       325    (5)       328    (5)
                                                 330    (5)       386    (6)       443    (7)       502    (8)       564    (9)
                                                 620    (10)      708    (12)      768    (13)      831    (14)      889    (15)
$DS_ASKLGCL_S     1             652    (11)      652    (11)
$DS_BGNMOD        1             72     (3)       72     (3)
$DS_BGNSUB        1             424    (7)       424    (7)       482    (8)       540    (9)       602    (10)      689    (12)
                                                 748    (13)      807    (14)      870    (15)
$DS_BGNTEST       1             383    (6)       383    (6)       649    (11)
$DS_BOPER         1             384    (6)       384    (6)
$DS_BREAK         1             437    (7)       437    (7)       496    (8)       558    (9)       614    (10)      624    (10)
                                                 702    (12)      762    (13)      825    (14)      883    (15)
$DS_CLRVEC_S      1             623    (10)      623    (10)      894    (15)
$DS_DSADEF        5             89     (3)       89     (3)
$DS_DSDEF         2             85     (3)       85     (3)
$DS_DSSDEF        8             86     (3)       86     (3)
$DS_ENDDATA       1             383    (6)       383    (6)       649    (11)
$DS_ENDMOD        1             905    (15)      905    (15)
$DS_ENDSUB        1             445    (7)       445    (7)       504    (8)       566    (9)       622    (10)      711    (12)
                                                 770    (13)      833    (14)      892    (15)
$DS_ENDTEST       1             624    (10)      624    (10)
$DS ENVDEF        2             72     (3)       72     (3)
$DS ERRHARD_S     1             315    (5)       315    (5)       318    (5)       320    (5)       324    (5)       327    (5)
```

```
                                                    329      (5)        442      (7)        501      (8)        563      (9)        619      (10)
                                                    707      (12)       767      (13)       830      (14)       888      (15)
$DS_EXIT               1          653      (11)     653      (11)
$DS_PAGE               1          357      (5)      357      (5)        391      (6)        447      (7)        506      (8)        568      (9)
                                                    626      (10)       656      (11)       713      (12)       772      (13)       835      (14)
$DS_PARDEF             1          88       (3)      88       (3)
$DS_PRINTF_S           1          385      (6)      385      (6)        390      (6)        434      (7)        493      (8)        555      (9)
                                                    612      (10)       655      (11)       699      (12)       759      (13)       822      (14)
                                                    880      (15)       881      (15)
$DS_PSLDEF             1          87       (3)      87       (3)
$DS_SBTTL              1          359      (6)      359      (6)        393      (7)        449      (8)        508      (9)        570      (10)
                                                    628      (11)       658      (12)       715      (13)       775      (14)       838      (15)
$DS_SCBDEF             3          84       (3)      84       (3)
$DS_SECDEF             1          91       (3)      91       (3)
$DS_SETVEC_S           1          387      (6)      387      (6)        651      (11)
$EQU                   1          88       (3)      72       (3)        85       (3)        87       (3)        88       (3)
$EQULS1                1          87       (3)      72       (3)        85       (3)        87       (3)
$EQULST                1          72       (3)      72       (3)        85       (3)        87       (3)
$GBLINI                2          72       (3)      72       (3)        85       (3)        87       (3)        88       (3)
$PSLDEF                2          87       (3)      87       (3)
$PUSHADR               1          315      (5)      315      (5)        318      (5)        320      (5)        324      (5)        327      (5)
                                                    329      (5)        442      (7)        501      (8)        563      (9)        619      (10)
                                                    652      (11)       707      (12)       767      (13)       830      (14)       888      (15)
$VIELD                 1          72       (3)      72       (3)        87       (3)        88       (3)
$VIELD1                1          88       (3)      72       (3)        87       (3)        88       (3)
ECKAX_SECDEF           1          91       (3)      91       (3)

                                               +-------------------------+
                                               ! Performance indicators !
                                               +-------------------------+

Phase                        Page faults    CPU Time       Elapsed Time
-----                        -----------    --------       ------------
Initialization                       116    00:00:00.34    00:00:03.32
Command processing                   909    00:00:00.97    00:00:02.34
Pass 1                               929    00:00:17.05    00:00:35.56
Symbol table sort                      0    00:00:00.64    00:00:00.90
Pass 2                                18    00:00:04.45    00:00:08.13
Symbol table output                   11    00:00:00.30    00:00:00.97
Psect synopsis output                  2    00:00:00.03    00:00:00.03
Cross-reference output                 6    00:00:01.51    00:00:02.14
Assembler run totals                1993    00:00:25.29    00:00:53.41

The working set limit was 3062 pages.
216678 bytes (424 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 458 non-local and 35 local symbols.
907 source lines were read in Pass 1, producing 34 object records in Pass 2.
105 pages of virtual memory were used to define 45 macros.
```

```
                                            +--------------------------+
                                            ! Macro library statistics !
                                            +--------------------------+

Macro library name                           Macros defined
------------------                           --------------
DISK$DIAGPACK02:[TAYLOR.ECKAX]ECKAX.MLB;1                1
SYS$COMMON:[SYSLIB]DIAG.MLB;955                         39
SYS$COMMON:[SYSLIB]STARLET.MLB;1                         7
TOTALS (all libraries)                                  47

789 GETS were required to define 47 macros.

There were no errors, warnings or information messages.

MACRO/LIS=ECKAX3.LIS/CROSS ECKAX3
```

Table of contents

```
   (1)        1        ECKAX4: PROCESSOR HALT
   (2)       69        DECLARATIONS     .LIST   MEB
   (3)      185        INSTRUCION MESSAGES
   (4)      225        ERROR MESSAGES
   (5)      346        DATA TABLES
   (6)      430        SUBROUTINES
   (7)      598        TEST 4:  PROCESSOR HALTS ARCHITECTURALLY DEFINED
   (8)      623        SUBTEST 4.1:  HALT instruction in Kernel Mode
   (9)      707        SUBTEST 4.2:  Interrupt Stack invalid
   (10)     870        SUBTEST 4.3:  IS SP points to NXM
   (11)     991        SUBTEST 4.4:  CHMU instruction with PSL<IS>=1
   (12)    1107        SUBTEST 4.5:  CHMS instruction with PSL<IS>=1
   (13)    1226        SUBTEST 4.6:  CHME instruction with PSL<IS>=1
   (14)    1344        SUBTEST 4.7:  CHMK instruction with PSL<IS>=1
   (15)    1467        TEST 5:  PROCESSOR HALTS ARCH. UNDEFINED
   (16)    1501        SUBTEST 5.1:  CHMU inst. with vector service on IS
   (17)    1587        SUBTEST 5.2:  CHMS inst. with vector service on IS
   (18)    1673        SUBTEST 5.3:  CHME inst. with vector service on IS
   (19)    1759        SUBTEST 5.4:  CHMK inst. with vector service on IS
   (20)    1850        SUBTEST 5.5:  Invalid vector HALT
```

```
0000          .SBTTL  ECKAX4: PROCESSOR HALT
0000          .TITLE  ECKAX4  VAX-11/750 Specific CPU Cluster Exerciser
0000          .IDENT  /5.2/
0000
0000    ;
0000    ; Copyright (C) 1980,1983,1986
0000    ; Digital Equipment Corporation, Maynard, Massachusetts 01754
0000    ;
0000    ; This software is furnished  under a license for use only on a single
0000    ; computer  system and  may be  copied only with  the inclusion of the
0000    ; above copyright notice.  This software, or any other copies thereof,
0000    ; may not be provided or  otherwise made available to any other person
0000    ; except for use on such system and to one who agrees to these license
0000    ; terms.  Title to and  ownership of the  software  shall at all times
0000    ; remain in DEC.
0000    ;
0000    ; The information in this software is subject to change without notice
0000    ; and should  not be construed  as a commitment  by Digital  Equipment
0000    ; Corporation.
0000    ;
0000    ; Dec assumes  no  responsibility  for  the use or  reliability of its
0000    ; software on equipment which is not supplied by DEC.
0000    ;
0000
0000    ;++
0000    ; Facility:     VAX Architecture Diagnostic.
0000    ;
0000    ; Abstract:     This module checks the various ways the VAX-11/750 processor
0000    ;               can be halted in an ARCHITECTURALLY DEFINED and an ARCHITEC-
0000    ;               TURALLY UNDEFINED manner. These ways
0000    ;               include executing a HALT instruction in kernel mode, attempting
0000    ;               certain references to an invalid interrupt stack, executing
0000    ;               a change mode instruction when the IS bit of the PSL is set,
0000    ;               a change mode instruction with vertor service on IS and invalid
0000    ;               vector Halt.
0000    ;
0000    ; Environment:  VAX Diagnostic Supervisor.
0000    ;
0000    ; Author:       Fred Roemer     16-Jun-80       Version 1.0
0000    ;
0000    ;       NO CHANGES TO THIS MODULE FOR VERSION 3.3 (DOCUMENT CHANGE ONLY)
0000    ;
0000    ;
0000    ;       Dan Milleville  19-Mar-82               Version 3.4
0000    ;       Changed Test 3 exit code to accomodate temporarily a hardware problem.
0000    ;
0000    ;       Dave Shull      30-Nov-82               Version 4.0
0000    ;       Added tests 13 and 14 for coverage of PCS module.
0000    ;
0000    ;       Tai-Chun Pan    04-Mar-83               Version 5.0
0000    ;       Added test 5 for coverage of Processor Halts Architecture Undefined.
0000    ;       Added tests 11 and 12 for coverage of new ROM with MRSP of TU58.
0000    ;
0000    ;       Dan Milleville  21-Nov-83               Version 5.1
0000    ;       Fixed error in timer routine (didn't turn off timer).
```

```
     0000     56 ;
     0000     57 ;        Wayne R. Taylor February 1986           Version 5.2
     0000     58 ;        Redefined IPR's to jive with starlet.
     0000     59 ;        Modified the code for problem report Eckax5 where:
     0000     60 ;                Tests 6 & 13 tests with cache off.
     0000     61 ;                Subtest 1 does not check bit 0 of the memory under test.
     0000     62 ;                Subtest 4 tests row 0, 1, 2, but not row 3.
     0000     63 ;        Indicate to user if the operator flag is not set for tests 2 through 5.
     0000     64 ;        Modifications were supplied by the submitter.
     0000     65 ;        Search on 5.2.
     0000     66 ;
     0000     67 ;--
```

```
                  0000     69          .SBTTL  DECLARATIONS    .LIST   MEB
                  0000     70          .NLIST  CND
              00000000     71          .PSECT ECKAXD,  LONG
                  0000     72          .LIBRARY        "SYS$LIBRARY:DIAG' ; VAX family diagnostic library.
                  0000     73          $DS_BGNMOD      CEP_REPAIR, TN=4
                  0000        ;;;      Macro-32 Diagnostic macro library Version 8.3 'DIAG.MLB (245)'
                  0000     74
                  0000     75 ;
                  0000     76 ; Include files:
                  0000     77 ;
                  0000     78
                  0000     79          .LIBRARY        'ECKAX"          ; CPU Cluster Exerciser library.
                  0000     80
                  0000     81 ;
                  0000     82 ; Equated symbols:
                  0000     83 ;
                  0000     87
                  0000     88          $DS_SCBDEF
                  0000     89          $DS_DSDEF
                  0000     90          $DS_ERRDEF
                  0000     91          $DS_DSSDEF
                  0000     92          $DS_PSLDEF
                  0000     93          $DS_DSADEF
                  0000     94
                  0000     95          ECKAX_SECDEF
                  0000     96
                  0000     97
      00000004    0000     98 B_ANSWER:       .BLKL                        ; used to storage answer (for asklgcl)
      00000008    0004     99 RENTRY_POINT:   .BLKL                        ; used to store rentry address
      0000000C    0008    100 L_PAGESTRT:     .BLKL                        ; first of 2 longword array to store
      00000010    000C    101 L_PAGEND:       .BLKL                        ; page boundary addresses to setprot
      00000014    0010    102 L_PAGEZERO:     .BLKL                        ; first of 2 longword array to store
      00000018    0014    103 L_PAGE0END:     .BLKL                        ; page 0 boundary addresses to setprot
                  0018    104
      0000001C    0018    105 L_COUNT:        .BLKL                        ; used as a counter
      00000020    001C    106 L_ADDRESS:      .BLKL                        ; used to temporary store addresses
      00000024    0020    107 L_CUR_ISP:      .BLKL                        ; used to store initial ISP
      00000028    0024    108 L_EXP_MOD:      .BLKL                        ; temp storage of expected mode
      0000002C    0028    109 L_EXP_STACK:    .BLKL                        ; temp storage of expected stack
      00000030    002C    110 L_OLDPROT:      .BLKL                        ; byte to store old page protection
      00000034    0030    111 L_TEMP:         .BLKL                        ; temporary storage
      00000038    0034    112 L_CUR_SCBB:     .BLKL                        ; location to store SCBB
      0000003C    0038    113 L_KA_FLAGS:     .BLKL
                  003C    114
      00000040    003C    115 L_FLAGS:        .BLKL                        ; contains the following flags
                  0040    116                                              ; bit positions and masks
      00000000    0040    117 KA_UNIT_NO=0
      00000000    0040    118 V_READY=0                                    ; ready to execute flag
      00000001    0040    119 M_READY=1a0                                  ;
      00000001    0040    120 V_DONE=1                                     ; restart done flag
      00000002    0040    121 M_DONE=1a1                                   ;
      00000002    0040    122 V_EXECUTE_ERR=2                              ; execution error flag
      00000004    0040    123 M_EXECUTE_ERR=1a2                            ;
      00000003    0040    124 V_CHMX_ERR=3                                 ; change mode error
      00000008    0040    125 M_CHMX_ERR=1a3                               ;
```

```
                  00000004  0040    126 V_XFC_ERR=4                                        ; XFC error
                  00000010  0040    127 M_XFC_ERR=1@4                                      ;
                  00000005  0040    128 V_TIMER=5                                          ; timer interrupt flag
                  00000020  0040    129 M_TIMER=1@5                                        ;
                  00000006  0040    130 V_BAD_MODE=6                                       ; bad mode, abort test
                  00000040  0040    131 M_BAD_MODE=1@6                                     ;
                  00000007  0040    132 V_BAD_STK=7                                        ; stack bad, abort test
                  00000080  0040    133 M_BAD_STK=1@7                                      ;
                  00000008  0040    134 V_FATAL_ERR=8                                      ; flag fatal error
                  00000100  0040    135 M_FATAL_ERR=1@8                                    ;
                  00000009  0040    136 V_PSL_ERR=9                                        ; flag PSL error
                  00000200  0040    137 M_PSL_ERR=1@9                                      ;
                  0000000A  0040    138 V_GPR_ERR=10                                       ; flag GPR error
                  00000400  0040    139 M_GPR_ERR=1@10                                     ;
                  0000000B  0040    140 V_XFC=11                                           ; expected XFC
                  00000800  0040    141 M_XFC=1@11                                         ;
                  0000000C  0040    142 V_ERROR_FLAG=12                                    ; flag any error
                  00001000  0040    143 M_ERROR_FLAG=1@12                                  ;
                  0000000D  0040    144 V_KSP_ERR=13                                       ; flag KSP error
                  00002000  0040    145 M_KSP_ERR=1@13                                     ;
                  0000000E  0040    146 V_ISP_ERR=14                                       ; flag ISP error
                  00004000  0040    147 M_ISP_ERR=1@14                                     ;
                  0000000F  0040    148 V_SP_ERR=15                                        ; flag SP error
                  00008000  0040    149 M_SP_ERR=1@15                                      ;
                  00000010  0040    150 V_ISTK=16                                          ; flag interrupt stack
                  00010000  0040    151 M_ISTK=1@16                                        ;
                  00500200  0040    152 L_NXM_ADR=^X500200                                 ; address of nonexist memory
                  00E00000  0040    153 L_BNXM_ADR=^XE00000                                ; big address
                  00000019  0040    154 V_USR_WCS=25
                            0040    155
                  00000000  0040    156 CODE_00=^X0                                        ; halt code equates
                  00000002  0040    157 CODE_02=^X2
                  00000004  0040    158 CODE_04=^X4
                  00000005  0040    159 CODE_05=^X5
                  00000006  0040    160 CODE_06=^X6
                  00000007  0040    161 CODE_07=^X7
                  00000008  0040    162 CODE_08=^X8
                  0000000A  0040    163 CODE_0A=^XA
                  0000000B  0040    164 CODE_0B=^XB
                  0000000C  0040    165 CODE_0C=^XC
                  0000000F  0040    166 CODE_0F=^XF
                  00000011  0040    167 CODE_11=^X11
                  00000012  0040    168 CODE_12=^X12
                  00000013  0040    169 CODE_13=^X13
                  00000014  0040    170 CODE_14=^X14
                  00000015  0040    171 CODE_15=^X15
                  00000016  0040    172 CODE_16=^X16
                  000000FF  0040    173 CODE_FF=^XFF
                            0040    174
                  00000080  0040    175 BUFSIZ=128
                            0040    176
                  00000100  0040    177 RESTART=^X100                                      ; label for memory loc 100
     0000003C'9F    02   C8 0040    178 JUMP:   BISL2   #M_DONE,@#L_FLAGS                  ; code to set done and go to routine
         00000C6A'9F     17 0047    179             JMP     @#READ_GPR                         ; to read all GPR's after a halt
                            004D    180
```

```
                                     004D   185 .SBTTL INSTRUCION MESSAGES
                                     004D   186
                                     004D   187 ;---------------------------------------------------------------------------
                                     004D   188 ; 5.2
                                     004D   189
                                     004D   190 T_NO_OPER:
6C 61 75 6E 61 4D 20 2F 21 2F 21 00' 004D   191         .ASCIC  "!/!/ Manual Section:  Tests 2 through 5 require operator interven -
cont> tion.!/"-
54 20 20 3A 6E 6F 69 74 63 65 53 20  0059
75 6F 72 68 74 20 32 20 73 74 73 65  0065
65 72 69 75 71 65 72 20 35 20 68 67  0071
6E 69 20 72 6F 74 61 72 65 70 6F 20  007D
21 2E 6E 6F 69 74 6E 65 76 72 65 74  0089
65 70 6F 20 65 68 74 20 74 65 53 2F  0095
74 20 67 61 6C 66 20 72 6F 74 61 72  00A1
65 74 20 65 74 75 63 65 78 65 20 6F  00AD
67 75 6F 72 68 74 20 32 20 73 74 73  00B9
               2F 21 20 2E 35 20 68  00C5
                                     00CC   192                  Set the operator flag to execute tests 2 through 5. !/"-
20 74 72 61 74 73 65 72 20 64 6E 61  00CC   193                  and restart the sections. "
73 6E 6F 69 74 63 65 73 20 65 68 74  00D8
                              20 2E  00E4
                                 98  004D
                                     00E6   194 ;------------------------------------------------------------------------------
                                     00E6   195 T_DBL_MCHK_ERR:
63 6E 65 72 65 66 65 72 20 61 20 00' 00E6   196         .ASCIC    a reference to NXM with vector pointing to NXM location.!/!/"
74 69 77 20 4D 58 4E 20 6F 74 20 65  00F2
69 6F 70 20 72 6F 74 63 65 76 20 68  00FE
4D 58 4E 20 6F 74 20 67 6E 69 74 6E  010A
2F 21 2E 6E 6F 69 74 61 63 6F 6C 20  0116
                              2F 21  0122
                                 3D  00E6
                                     0124   197 ;------------------------------------------------------------------------------
                                     0124   198 T_NXM_WCS_ERR:
72 65 66 73 6E 61 72 74 20 61 20 00' 0124   199         .ASCIC    a transfer to user WCS which has not been loaded.!/!/"
53 43 57 20 72 65 73 75 20 6F 74 20  0130
6E 20 73 61 68 20 68 63 69 68 77 20  013C
64 61 6F 6C 20 6E 65 65 62 20 74 6F  0148
               2F 21 2F 21 2E 64 65  0154
                                 36  0124
                                     015B   200 ;---------------------------------------------- ------------------------------
                                     015B   201 T_SCBBNXM_ERR:
73 6E 69 20 43 46 58 20 6E 61 20 00' 015B   202         .ASCIC  ' an XFC inst with SCBB pointing to non-existent memory.!/!/
20 42 42 43 53 20 68 74 69 77 20 74  0167
20 6F 74 20 67 6E 69 74 6E 69 6F 70  0173
74 6E 65 74 73 69 78 65 2D 6E 6F 6E  017F
2F 21 2F 21 2E 79 72 6F 6D 65 6D 20  018B
                                 3B  015B
                                     0197   203 ;------------------------------------------------------------------------------
                                     0197   204 T_REST_OFF:
4F 54 55 41 20 65 68 74 20 73 49 00' 0197   205         .ASCIC  'Is the AUTO RESTART switch in the off position?"
69 77 73 20 54 52 41 54 53 45 52 20  01A3
6F 20 65 68 74 20 6E 69 20 68 63 74  01AF
3F 6E 6F 69 74 69 73 6F 70 20 66 66  01BB
                                 2F  0197
                                     01C7   206 ;---------------------------------------------------------------------------
```

```
                                          01C7    207 T_EXITING:
4C 41 48 20 67 6E 69 74 69 78 45 00'  01C7    208         .ASCIC  'Exiting HALT test.!/"
         2F 21 2E 74 73 65 74 20 54  01D3
                                  14  01C7
                                      01DC    209
                                      01DC    210 ;----------------------------------------------------------------------------
                                      01DC    211 T_INSTRUCTIONS:
65 70 78 65 20 54 4C 41 48 2F 21 00'  01DC    212         .ASCIC  '!/HALT expected with following printout:!/'-
6F 66 20 68 74 69 77 20 64 65 74 63  01E8
6E 69 72 70 20 67 6E 69 77 6F 6C 6C  01F4
               2F 21 3A 74 75 6F 74  0200
   2F 21 20 42 58 21 20 20 4C 58 21  0207    213                 !XL  !XB !/
                                  35  01DC
                                      0212    214
                                      0212    215 ;---------------------------------------------------------------------------
                                      0212    216 T_CONTINUE:
65 75 6E 69 74 6E 6F 63 20 6F 54 00'  0212    217         .ASCIC   To continue from console mode do the following:!/ -
6C 6F 73 6E 6F 63 20 6D 6F 72 66 20  021E
68 74 20 6F 64 20 65 64 6F 6D 20 65  022A
3A 67 6E 69 77 6F 6C 6C 6F 66 20 65  0236
                              2F 21  0242
43 3C 20 30 30 31 20 46 20 47 2F 44  0244    218                 'D/G F 100 <CR> and then C <CR>'
20 6E 65 68 74 20 64 6E 61 20 3E 52  0250
            3E 52 43 3C 20 43        025C
                                  4F  0212
                                      0262    219 ;---------------------------------------------------------------------------
                                      0262    220 T_WARNING:
20 3A 47 4E 49 4E 52 41 57 2F 21 00'  0262    221         .ASCIC  "!/WARNING: The front panel switch must be in the HALT!/ -
61 70 20 74 6E 6F 72 66 20 65 68 54  026E
6D 20 68 63 74 69 77 73 20 6C 65 6E  027A
68 74 20 6E 69 20 65 62 20 74 73 75  0286
            2F 21 54 4C 41 48 20 65  0292
20 73 69 68 74 20 72 6F 66 20 20 09  029A    222                 '          for this test to operate properly.!/
72 65 70 6F 20 6F 74 20 74 73 65 74  02A6
79 6C 72 65 70 6F 72 70 20 65 74 61  02B2
                           2F 21 2E  02BE
                                  5E  0262
                                      02C1    223 ;---------------------------------------------------------------------------
```

```
                                                    02C1     225 .SBTTL  ERROR MESSAGES
                                                    02C1     226 ;------------------------------------------------------------------
                                                    02C1     227 T_HALT_MESS:
20 64 65 74 70 6D 65 74 74 41 20 00'  02C1     228         .ASCIC  ' Attempted to halt the processor by executing!/'
20 65 68 74 20 74 6C 61 68 20 6F 74   02CD
79 62 20 72 6F 73 73 65 63 6F 72 70   02D9
2F 21 67 6E 69 74 75 63 65 78 65 20   02E5
                                   2F  02C1
                                       02F1     229 ;------------------------------------------------------------------
                                       02F1     230 T_HALT_ERR:
6F 63 20 6F 72 63 61 6D 20 61 20 00'  02F1     231         .ASCIC  ' a macro code HALT instruction.!/!/'
74 73 6E 69 20 54 4C 41 48 20 65 64   02FD
2F 21 2F 21 2E 6E 6F 69 74 63 75 72   0309
                                   23  02F1
                                       0315     232 ;------------------------------------------------------------------
                                       0315     233 T_INVLDIS_ERR:
73 6E 69 20 43 46 58 20 6E 61 20 00'  0315     234         .ASCIC    an XFC inst with the IS pointing to a protected page.!/!/'
49 20 65 68 74 20 68 74 69 77 20 74   0321
74 20 67 6E 69 74 6E 69 6F 70 20 53   032D
65 74 63 65 74 6F 72 70 20 61 20 6F   0339
   2F 21 2F 21 2E 65 67 61 70 20 64   0345
                                   3A  0315
                                       0350     235 ;------------------------------------------------------------------
                                       0350     236 T_NXMIS_ERR:
73 6E 69 20 43 46 58 20 6E 61 20 00'  0350     237         .ASCIC    an XFC inst with the IS pointing to nonexistent memory.!/!/'
49 20 65 68 74 20 68 74 69 77 20 74   035C
74 20 67 6E 69 74 6E 69 6F 70 20 53   0368
6E 65 74 73 69 78 65 6E 6F 6E 20 6F   0374
21 2F 21 2E 79 72 6F 6D 65 6D 20 74   0380
                                   2F  038C
                                   3C  0350
                                       038D     238 ;------------------------------------------------------------------
                                       038D     239 T_CHMU_ERR:
73 6E 69 20 55 4D 48 43 20 61 20 00'  038D     240         .ASCIC    a CHMU instruction with PSL<IS>=1.!/!/
74 69 77 20 6E 6F 69 74 63 75 72 74   0399
2E 31 3D 3E 53 49 3C 4C 53 50 20 68   03A5
                        2F 21 2F 21   03B1
                                   27  038D
                                       03B5     241 ;------------------------------------------------------------------
                                       03B5     242 T_CHMS_ERR:
73 6E 69 20 53 4D 48 43 20 61 20 00'  03B5     243         .ASCIC    a CHMS instruction with PSL<IS>=1.!/!/
74 69 77 20 6E 6F 69 74 63 75 72 74   03C1
2E 31 3D 3E 53 49 3C 4C 53 50 20 68   03CD
                        2F 21 2F 21   03D9
                                   27  03B5
                                       03DD     244 ;------------------------------------------------------------------
                                       03DD     245 T_CHME_ERR:
73 6E 69 20 45 4D 48 43 20 61 20 00'  03DD     246         .ASCIC    a CHME instruction with PSL<IS>=1.!/!/
74 69 77 20 6E 6F 69 74 63 75 72 74   03E9
2E 31 3D 3E 53 49 3C 4C 53 50 20 68   03F5
                        2F 21 2F 21   0401
                                   27  03DD
                                       0405     247 ;------------------------------------------------------------------
                                       0405     248 T_CHMK_ERR:
73 6E 69 20 4B 4D 48 43 20 61 20 00'  0405     249         .ASCIC    a CHMK instruction with PSL<IS>=1.!/!/'
```

```
74 69 77 20 6E 6F 69 74 63 75 72 74  0411
2E 31 3D 3E 53 49 3C 4C 53 50 20 68  041D
                        2F 21 2F 21  0429
                                 27  0405
                                     042D     250 ;-----------------------------------------------------------------------
                                     042D     251 T_SCB_ERR:
73 6E 69 20 43 46 58 20 6E 61 20 00' 042D     252          .ASCIC    an XFC inst with the SCB vector entry <1:0> = 3.!/!/
53 20 65 68 74 20 68 74 69 77 20 74  0439
6E 65 20 72 6F 74 63 65 76 20 42 43  0445
20 3D 20 3E 30 3A 31 3C 20 79 72 74  0451
                  2F 21 2F 21 2E 33  045D
                                 35  042D
                                     0463     253 ;-----------------------------------------------------------------------
                                     0463     254 T_WCS_ERR:
73 6E 69 20 43 46 58 20 6E 61 20 00' 0463     255          .ASCIC   ' an XFC inst with SCB vector entry <1:0>=2!/' -
76 20 42 43 53 20 68 74 69 77 20 74  046F
20 79 72 74 6E 65 20 72 6F 74 63 65  047B
         2F 21 32 3D 3E 30 3A 31 3C  0487
73 69 64 20 53 43 57 20 64 6E 61 20  0490     256                   ' and WCS disabled or not present!/!/.'
74 6F 6E 20 72 6F 20 64 65 6C 62 61  049C
2F 21 2F 21 74 6E 65 73 65 72 70 20  04A8
                                 2E  04B4
                                 51  0463
                                     04B5     257 ;-----------------------------------------------------------------------
                                     04B5     258 T_CHMUSCB_ERR:
73 6E 69 20 55 4D 48 43 20 61 20 00' 04B5     259          .ASCIC    a CHMU inst with the SCB vector entry <1:0> not = 0 .!/!/
53 20 65 68 74 20 68 74 69 77 20 74  04C1
6E 65 20 72 6F 74 63 65 76 20 42 43  04CD
6F 6E 20 3E 30 3A 31 3C 20 79 72 74  04D9
   2F 21 2F 21 2E 20 30 20 3D 20 74  04E5
                                 3A  04B5
                                     04F0     260 ;-----------------------------------------------------------------------
                                     04F0     261 T_CHMSSCB_ERR:
73 6E 69 20 53 4D 48 43 20 61 20 00' 04F0     262          .ASCIC    a CHMS inst with the SCB vector entry <1:0> not = 0 .!/!/'
53 20 65 68 74 20 68 74 69 77 20 74  04FC
6E 65 20 72 6F 74 63 65 76 20 42 43  0508
6F 6E 20 3E 30 3A 31 3C 20 79 72 74  0514
   2F 21 2F 21 2E 20 30 20 3D 20 74  0520
                                 3A  04F0
                                     052B     263 ;-----------------------------------------------------------------------
                                     052B     264 T_CHMESCB_ERR:
73 6E 69 20 45 4D 48 43 20 61 20 00' 052B     265          .ASCIC    a CHME inst with the SCB vector entry <1:0> not = 0 .!/!/'
53 20 65 68 74 20 68 74 69 77 20 74  0537
6E 65 20 72 6F 74 63 65 76 20 42 43  0543
6F 6E 20 3E 30 3A 31 3C 20 79 72 74  054F
   2F 21 2F 21 2E 20 30 20 3D 20 74  055B
                                 3A  052B
                                     0566     266 ;-----------------------------------------------------------------------
                                     0566     267 T_CHMKSCB_ERR:
73 6E 69 20 4B 4D 48 43 20 61 20 00' 0566     268          .ASCIC    a CHMK inst with the SCB vector entry <1:0> not = 0 .!/!/
53 20 65 68 74 20 68 74 69 77 20 74  0572
6E 65 20 72 6F 74 63 65 76 20 42 43  057E
6F 6E 20 3E 30 3A 31 3C 20 79 72 74  058A
   2F 21 2F 21 2E 20 30 20 3D 20 74  0596
                                 3A  0566
```

```
                                      05A1   269 ;--------------------------------------------------------------------
                                      05A1   270 T_NOT_READY:
72 74 73 6E 69 20 74 73 65 54 20 00'  05A1   271           .ASCIC   ' Test instruction was not ready to be executed.!/
6E 20 73 61 77 20 6E 6F 69 74 63 75   05AD
20 6F 74 20 79 64 61 65 72 20 74 6F   05B9
2E 64 65 74 75 63 65 78 65 20 65 62   05C5
                               2F 21  05D1
                                  31  05A1
                                      05D3   272 ;--------------------------------------------------------------------
                                      05D3   273 T_NOT_DONE:
6C 66 20 74 72 61 74 73 65 52 20 00'  05D3   274           .ASCIC   ' Restart flag was not set while restarting.!/'
73 20 74 6F 6E 20 73 61 77 20 67 61   05DF
73 65 72 20 65 6C 69 68 77 20 74 65   05EB
      2F 21 2E 67 6E 69 74 72 61 74   05F7
                                  2D  05D3
                                      0601   275 ;------------------------------------------------------- ------------
                                      0601   276 T_EXECUTE:
72 74 73 6E 69 20 74 73 65 54 20 00'  0601   277           .ASCIC   ' Test instruction execution error.!/'
75 63 65 78 65 20 6E 6F 69 74 63 75   060D
21 2E 72 6F 72 72 65 20 6E 6F 69 74   0619
                                  2F  0625
                                  24  0601
                                      0626   278 ;--------------------------------------------------------------------
                                      0626   279 T_XFC:
75 72 74 73 6E 69 20 43 46 58 20 00'  0626   280           .ASCIC   ' XFC instruction executed instead of causing halt.!/
74 75 63 65 78 65 20 6E 6F 69 74 63   0632
6F 20 64 61 65 74 73 6E 69 20 64 65   063E
61 68 20 67 6E 69 73 75 61 63 20 66   064A
                     2F 21 2E 74 6C   0656
                                  34  0626
                                      065B   281 ;--------------------------------------------------------------------
                                      065B   282 T_CHMX_ERR:
72 74 73 6E 69 20 78 4D 48 43 20 00'  065B   283           .ASCIC     CHMx instruction executed instead of causing halt.!/
75 63 65 78 65 20 6E 6F 69 74 63 75   0667
20 64 61 65 74 73 6E 69 20 64 65 74   0673
68 20 67 6E 69 73 75 61 63 20 66 6F   067F
                  2F 21 2E 74 6C 61   068B
                                  35  065B
                                      0691   284 ;--------------------------------------------------------------------
                                      0691   285 T_TIMER:
64 65 74 63 65 70 78 65 6E 55 20 00'  0691   286           .ASCIC   ' Unexpected interval timer interrupt.!/
69 74 20 6C 61 76 72 65 74 6E 69 20   069D
70 75 72 72 65 74 6E 69 20 72 65 6D   06A9
                        2F 21 2E 74   06B5
                                  27  0691
                                      06B9   287 ;--------------------------------------------------------------------
                                      06B9   288 T_KSP_ERR1:
63 6E 69 20 73 69 20 50 53 4B 20 00'  06B9   289           .ASCIC     KSP is incorrect  -   EXP = !XL  ACT = !XL!/
20 20 20 2D 20 20 74 63 65 72 72 6F   06C5
41 20 20 4C 58 21 20 3D 20 50 58 45   06D1
      2F 21 4C 58 21 20 3D 20 54 43   06DD
                                  2D  06B9
                                      06E7   290 ;--------------------------------------------------------------------
                                      06E7   291 T_KSP_ERR2:
20 20 20 2D 20 20 20 50 53 4B 20 00'  06E7   292           .ASCIC     KSP   -   EXP =(UNPREDICTABLE)   ACT = !XL!/
```

```
44 45 52 50 4E 55 28 3D 20 50 58 45  06F3
41 20 20 20 29 45 4C 42 41 54 43 49  06FF
      2F 21 4C 58 21 20 3D 20 54 43  070B
                                 2D  06E7
                                     0715   293 ;--------------------------------------------------------------------------------
                                     0715   294 T_ISP_ERR1:
63 6E 69 20 73 69 20 50 53 49 20 00' 0715   295         .ASCIC    ISP is incorrect  -   EXP = !XL  ACT = !XL!/'
20 20 20 2D 20 20 74 63 65 72 72 6F  0721
41 20 20 4C 58 21 20 3D 20 50 58 45  072D
      2F 21 4C 58 21 20 3D 20 54 43  0739
                                 2D  0715
                                     0743   296 ;--------------------------------------------------------------------------------
                                     0743   297 T_ISP_ERR2:
20 20 20 2D 20 20 20 50 53 49 20 00' 0743   298         .ASCIC    ISP   -   EXP =(UNPREDICTABLE)   ACT = !XL!/'
44 45 52 50 4E 55 28 3D 20 50 58 45  074F
41 20 20 20 29 45 4C 42 41 54 43 49  075B
      2F 21 4C 58 21 20 3D 20 54 43  0767
                                 2D  0743
                                     0771   299 ;--------------------------------------------------------------------------------
                                     0771   300 T_SP_ERR1:
6F 63 6E 69 20 73 69 20 50 53 20 00' 0771   301         .ASCIC    SP is incorrect   -   EXP = !XL  ACT = !XL!/
20 20 20 2D 20 20 20 74 63 65 72 72  077D
41 20 20 4C 58 21 20 3D 20 50 58 45  0789
      2F 21 4C 58 21 20 3D 20 54 43  0795
                                 2D  0771
                                     079F   302 ;--------------------------------------------------------------------------------
                                     079F   303 T_SP_ERR2:
2F 21 4C 58 21 20 3D 20 50 53 20 00' 079F   304         .ASCIC    " SP = !XL!/'
                                 0B  079F
                                     07AB   305 ;--------------------------------------------------------------------------------
                                     07AB   306 T_GPR_ERR:
69 20 61 74 61 64 20 64 61 42 20 00' 07AB   307         .ASCIC    Bad data in general purpose register:!/
75 70 20 6C 61 72 65 6E 65 67 20 6E  07B7
74 73 69 67 65 72 20 65 73 6F 70 72  07C3
                     2F 21 3A 72 65  07CF
                                 28  07AB
                                     07D4   308 ;--------------------------------------------------------------------------------
                                     07D4   309 T_BAD_DATA:
20 4C 58 21 20 3D 20 50 58 45 20 00' 07D4   310         .ASCIC    EXP = !XL    ACT = !XL    XOR= !XL !/
4C 58 21 20 3D 20 54 43 41 20 20 20  07E0
4C 58 21 20 3D 52 4F 58 20 20 20 20  07EC
                           2F 21 20  07F8
                                 26  07D4
                                     07FB   311 ;--------------------------------------------------------------------------------
                                     07FB   312 T_GOOD_DATA:
20 4C 58 21 20 3D 20 50 58 45 20 00' 07FB   313         .ASCIC    EXP = !XL    ACT = !XL !/
4C 58 21 20 3D 20 54 43 41 20 20 20  0807
                           2F 21 20  0813
                                 1A  07FB
                                     0816   314 ;--------------------------------------------------------------------------------
                                     0816   315 T_PAGE0PROT:
20 6F 74 20 65 6C 62 61 6E 55 20 00' 0816   316         .ASCIC    Unable to set proper protection on page 0.!/
70 20 72 65 70 6F 72 70 20 74 65 73  0822
6E 6F 20 6E 6F 69 74 63 65 74 6F 72  082E
      2F 21 2E 30 20 65 67 61 70 20  083A
```

```
                                 2D  0816
                                     0844     317 ;----------------------------------------------------------------------------
                                     0844     318 T_PAGEPROT:
20 6F 74 20 65 6C 62 61 6E 55 20 00' 0844     319           .ASCIC  ' Unable to set proper protection on buffer page.!/
70 20 72 65 70 6F 72 70 20 74 65 73  0850
6E 6F 20 6E 6F 69 74 63 65 74 6F 72  085C
65 67 61 70 20 72 65 66 66 75 62 20  0868
                           2F 21 2E  0874
                                 32  0844
                                     0877     320 ;-------------------------------------------------------------------- --------
                                     0877     321 T_NOT_ON_KS:
72 65 4B 20 6E 6F 20 74 6F 4E 20 00' 0877     322           .ASCIC  ' Not on Kernel Stack as expected.!/
73 61 20 6B 63 61 74 53 20 6C 65 6E  0883
2F 21 2E 64 65 74 63 65 70 78 65 20  088F
                                 23  0877
                                     089B     323 ;----------------------------------------------------------- ----------------
                                     089B     324 T_NOT_ON_IS:
74 6E 49 20 6E 6F 20 74 6F 4E 20 00' 089B     325           .ASCIC  ' Not on Interrupt Stack as expected.!/
6B 63 61 74 53 20 74 70 75 72 72 65  08A7
64 65 74 63 65 70 78 65 20 73 61 20  08B3
                           2F 21 2E  08BF
                                 26  089B
                                     08C2     326 ;------------------------------------------------------------- -------------
                                     08C2     327 T_EXP_PSL:
50 20 64 65 74 63 65 70 78 45 20 00' 08C2     328           .ASCIC  ' Expected PSL   =   !_!XL(X) ; !AC!/
21 5F 21 20 20 20 3D 20 20 20 4C 53  08CE
21 43 41 21 20 3B 20 29 58 28 4C 58  08DA
                                 2F  08E6
                                 24  08C2
                                     08E7     329 ;----------------------------------------------------------------------------
                                     08E7     330 T_ACT_PSL:
4C 53 50 20 6C 61 75 74 63 41 20 00' 08E7     331           .ASCIC  ' Actual PSL     =   !_!XL(X) ; !AC!/
21 5F 21 20 20 20 3D 20 20 20 20 20  08F3
21 43 41 21 20 3B 20 29 58 28 4C 58  08FF
                                 2F  090B
                                 24  08E7
                                     090C     332 ;----------------------------------------------------------------------------
                                     090C     333 T_PSL:
20 20 20 3D 20 20 20 4C 53 50 20 00' 090C     334           .ASCIC    PSL   =   !_!XL (X) ; !AC!/
20 3B 20 29 58 28 20 4C 58 21 5F 21  0918
                     2F 21 43 41 21  0924
                                 1C  090C
                                     0929     335 ;------- --------------------------------------------------------------------
                                     0929     336 T_GPR:
20 20 72 65 74 73 69 67 65 52 20 00' 0929     337           .ASCIC    Register  Expected    Actual     XOR!/
20 20 20 20 64 65 74 63 65 70 78 45  0935
58 20 20 20 20 20 6C 61 75 74 63 41  0941
                        2F 21 52 4F  094D
                                 27  0929
                                     0951     338 ;----------------------------------------------------------------------------
                                     0951     339 T_XOR:
4C 58 21 20 20 5F 21 43 41 21 20 00' 0951     340           .ASCIC    !AC!_  !XL   !XL   !XL!/'
4C 58 21 20 20 20 4C 58 21 20 20 20  095D
                              2F 21  0969
                                 19  0951
                                 096B   341 ;----------------------------------------------------------------------------
                                 096B   342 T_NOXOR:
4C 58 21 20 20 5F 21 43 41 21 20 00' 096B   343         .ASCIC  ' !AC!_  !XL   !XL!/"
         2F 21 4C 58 21 20 20 20 0977
                              13 096B
                                 097F   344 ;----------------------------------------------------------------------------
```

```
                                     097F   346 .SBTTL  DATA TABLES
                                     097F   347
                                     097F   348 ; Table for test 1
                                     097F   349
                           00000000  097F   350 DATA_1:          .LONG   ^X0             ; expected manual restart PSL
                                     0983   351
                           041F0000  0983   352 DATA_2:          .LONG   ^X41F0000       ; expected PSL
                                     0987   353
                           041F0008  0987   354 DATA_3:          .LONG   ^X41F0008       ; expected PSL
                                     098B   355
                           041F0000  098B   356 DATA_4:          .LONG   ^X41F0000       ; expected PSL
                                     098F   357
                           041F0000  098F   358 DATA_5:          .LONG   ^X41F0000       ; expected PSL
                                     0993   359
                           041F0000  0993   360 DATA_6:          .LONG   ^X41F0000       ; expected PSL
                                     0997   361
                           041F0000  0997   362 DATA_7:          .LONG   ^X41F0000       ; expected PSL
                                     099B   363
                                     099B   364
                                     099B   365
                                     099B   366 ERR_MESSAGE:
                                     099B   367         $DS_BGNMESSAGE
                               0000  099B                        .WORD   ^M<>            ; ENTRY MASK
              5E   00000084 8F    C2  099D   368         SUBL2    #BUFSIZ+4,SP            ; Reserve room on stack for cvtreg
                       52   5E    D0  09A4   369         MOVL     SP,R2                   ; Save starting address of buffer
                                     09A7   370         $DS_PRINTB_S    FORMAT=T_HALT_MESS
                       F916 CF    9F  09A7                       PUSHAB  T_HALT_MESS
              000100E0 9F   01    FB  09AB                       CALLS   #$$N, @#DS$PRINTB
                                     09B2   371         $DS_PRINTB_S    FORMAT=@ERR$_P1(AP)
                         14 BC    9F  09B2                       PUSHAB  @ERR$_P1(AP)
              000100E0 9F   01    FB  09B5                       CALLS   #$$N, @#DS$PRINTB
               0B F67B CF   00    E0  09BC   372         BBS      #V_READY,L_FLAGS,1$     ; br = instruction ready to execute
                                     09C2   373         $DS_PRINTB_S    FORMAT=T_NOT_READY
                       FBDB CF    9F  09C2                       PUSHAB  T_NOT_READY
              000100E0 9F   01    FB  09C6                       CALLS   #$$N, @#DS$PRINTB
               0B F66A CF   01    E0  09CD   374 1$:     BBS      #V_DONE,L_FLAGS,2$      ; br = normal restart
                                     09D3   375         $DS_PRINTB_S    FORMAT=T_NOT_DONE
                       FBFC CF    9F  09D3                       PUSHAB  T_NOT_DONE
              000100E0 9F   01    FB  09D7                       CALLS   #$$N, @#DS$PRINTB
               0B F659 CF   02    E1  09DE   376 2$:     BBC      #V_EXECUTE_ERR,L_FLAGS,3$ ; br = instruction executed
                                     09E4   377         $DS_PRINTB_S    FORMAT=T_EXECUTE
                       FC19 CF    9F  09E4                       PUSHAB  T_EXECUTE
              000100E0 9F   01    FB  09E8                       CALLS   #$$N, @#DS$PRINTB
               0B F648 CF   03    E1  09EF   378 3$:     BBC      #V_CHMX_ERR,L_FLAGS,4$ ; br = no CHMx error
                                     09F5   379         $DS_PRINTB_S    FORMAT=T_CHMX_ERR
                       FC62 CF    9F  09F5                       PUSHAB  T_CHMX_ERR
              000100E0 9F   01    FB  09F9                       CALLS   #$$N, @#DS$PRINTB
               0B F637 CF   04    E1  0A00   380 4$:     BBC      #V_XFC_ERR,L_FLAGS,5$   ; br = no XFC error
                                     0A06   381         $DS_PRINTB_S    FORMAT=T_XFC
                       FC1C CF    9F  0A06                       PUSHAB  T_XFC
              000100E0 9F   01    FB  0A0A                       CALLS   #$$N, @#DS$PRINTB
               1F F626 CF   10    E1  0A11   382 5$:     BBC      #V_ISTK,L_FLAGS,6$      ; br = on kernel stack
               2A F620 CF   0D    E1  0A17   383         BBC      #V_KSP_ERR,L_FLAGS,7$   ; br = no KSP error
                                     0A1D   384         $DS_PRINTB_S    FORMAT=T_KSP_ERR1,P0=L_EXP_KSP,P1=L_ACT_KSP
                   00000000'EF    DD  0A1D                       PUSHL   L_ACT_KSP
```

```
      00000000'EF   DD  0A23                      PUSHL   L_EXP_KSP
           FC8C CF  9F  0A29                      PUSHAB  T_KSP_ERR1
   000100E0 9F   03  FB  0A2D                      CALLS   #$$N, @#DS$PRINTB
              11  11  0A34   385          BRB     7$
                      0A36   386 6$:      $DS_PRINTX_S    FORMAT=T_KSP_ERR2,P0=L_ACT_KSP
      00000000'EF   DD  0A36                      PUSHL   L_ACT_KSP
           FCA7 CF  9F  0A3C                      PUSHAB  T_KSP_ERR2
   000100E8 9F   02  FB  0A40                      CALLS   #$$N, @#DS$PRINTX
    1F F5F0 CF   10  E0  0A47   387 7$:      BBS     #V_ISTK,L_FLAGS,8$       ; br = on interrupt stack
    2A F5EA CF   0E  E1  0A4D   388          BBC     #V_ISP_ERR,L_FLAGS,9$    ; br = no ISP error
                      0A53   389          $DS_PRINTB_S    FORMAT=T_ISP_ERR1,P0=L_EXP_ISP,P1=L_ACT_ISP
      00000000'EF   DD  0A53                      PUSHL   L_ACT_ISP
      00000000'EF   DD  0A59                      PUSHL   L_EXP_ISP
           FCB2 CF  9F  0A5F                      PUSHAB  T_ISP_ERR1
   000100E0 9F   03  FB  0A63                      CALLS   #$$N, @#DS$PRINTB
              11  11  0A6A   390          BRB     9$                       ; skip extened error report
                      0A6C   391 8$:      $DS_PRINTX_S    FORMAT=T_ISP_ERR2,P0=L_ACT_ISP
      00000000'EF   DD  0A6C                      PUSHL   L_ACT_ISP
           FCCD CF  9F  0A72                      PUSHAB  T_ISP_ERR2
   000100E8 9F   02  FB  0A76                      CALLS   #$$N, @#DS$PRINTX
    19 F5BA CF   0F  E1  0A7D   392 9$:      BBC     #V_SP_ERR,L_FLAGS,10$    ; br = no SP error
                      0A83   393          $DS_PRINTB_S    FORMAT=T_SP_ERR1,P0=L_EXP_SP,P1=L_ACT_SP
      00000000'EF   DD  0A83                      PUSHL   L_ACT_SP
      00000000'EF   DD  0A89                      PUSHL   L_EXP_SP
           FCDE CF  9F  0A8F                      PUSHAB  T_SP_ERR1
   000100E0 9F   03  FB  0A93                      CALLS   #$$N, @#DS$PRINTB
              11  11  0A9A   394          BRB     11$                      ; skip extended error report
                      0A9C   395 10$:     $DS_PRINTX_S    FORMAT=T_SP_ERR2,P0=L_ACT_SP
      00000000'EF   DD  0A9C                      PUSHL   L_ACT_SP
           FCF9 CF  9F  0AA2                      PUSHAB  T_SP_ERR2
   000100E8 9F   02  FB  0AA6                      CALLS   #$$N, @#DS$PRINTX
    03 F58A CF   09  E0  0AAD   396 11$:     BBS     #V_PSL_ERR,L_FLAGS,12$   ; br = PSL error
            008A  31  0AB3   397          BRW     13$                      ; no PSL error
                      0AB6   398 12$:     $DS_CVTREG_S    MSB=#31,DATA=L_EXP_PSL,MNEADR=A_PSLMNE,-
                      0AB6   399                          STRBUF=(R2),MAXLEN=#BUFSIZ,-
                      0AB6   400                          V1=MODELIST,V2=MODELIST
              00  DD  0AB6                      PUSHL   #0
              00  DD  0AB8                      PUSHL   #0
              00  DD  0ABA                      PUSHL   #0
              00  DD  0ABC                      PUSHL   #0
      00000000'EF   DF  0ABE                      PUSHAL  MODELIST
      00000000'EF   DF  0AC4                      PUSHAL  MODELIST
      00000080 8F   DD  0ACA                      PUSHL   #BUFSIZ
              62  9F  0AD0                      PUSHAB  (R2)
      00000000'EF   9F  0AD2                      PUSHAB  A_PSLMNE
      00000000'EF   DD  0AD8                      PUSHL   L_EXP_PSL
              1F  DD  0ADE                      PUSHL   #31
   000100B0 9F   0B  FB  0AE0                      CALLS   #11, @#DS$CVTREG
                      0AE7   401          $DS_PRINTB_S    FORMAT=T_EXP_PSL,P0=L_EXP_PSL,P1=R2
              52  DD  0AE7                      PUSHL   R2
      00000000'EF   DD  0AE9                      PUSHL   L_EXP_PSL
           FDCF CF  9F  0AEF                      PUSHAB  T_EXP_PSL
   000100E0 9F   03  FB  0AF3                      CALLS   #$$N, @#DS$PRINTB
                      0AFA   402          $DS_CVTREG_S    MSB=#31,DATA=L_ACT_PSL,MNEADR=A_PSLMNE,-
                      0AFA   403                          STRBUF=(R2),MAXLEN=#BUFSIZ,-
```

```
                          0AFA   404                        V1=MODELIST,V2=MODELIST
                 00   DD  0AFA                     PUSHL    #0
                 00   DD  0AFC                     PUSHL    #0
                 00   DD  0AFE                     PUSHL    #0
                 00   DD  0B00                     PUSHL    #0
        00000000'EF   DF  0B02                     PUSHAL   MODELIST
        00000000'EF   DF  0B08                     PUSHAL   MODELIST
        00000080 8F   DD  0B0E                     PUSHL    #BUFSIZ
                 62   9F  0B14                     PUSHAB   (R2)
        00000000'EF   9F  0B16                     PUSHAB   A_PSLMNE
        00000000'EF   DD  0B1C                     PUSHL    L_ACT_PSL
                 1F   DD  0B22                     PUSHL    #31
   000100B0 9F   0B   FB  0B24                     CALLS    #11, a#DS$CVTREG
                          0B2B   405           $DS_PRINTB_S     FORMAT=T_ACT_PSL,P0=L_ACT_PSL,P1=R2
                 52   DD  0B2B                     PUSHL    R2
        00000000'EF   DD  0B2D                     PUSHL    L_ACT_PSL
            FDB0 CF   9F  0B33                     PUSHAB   T_ACT_PSL
   000100E0 9F   03   FB  0B37                     CALLS    #$$N, a#DS$PRINTB
                 44   11  0B3E   406           BRB      14$
                          0B40   407 13$:      $DS_CVTREG_S     MSB=#31,DATA=L_EXP_PSL,MNEADR=A_PSLMNE,-
                          0B40   408                        STRBUF=(R2),MAXLEN=#BUFSIZ,-
                          0B40   409                        V1=MODELIST,V2=MODELIST
                 00   DD  0B40                     PUSHL    #0
                 00   DD  0B42                     PUSHL    #0
                 00   DD  0B44                     PUSHL    #0
                 00   DD  0B46                     PUSHL    #0
        00000000'EF   DF  0B48                     PUSHAL   MODELIST
        00000000'EF   DF  0B4E                     PUSHAL   MODELIST
        00000080 8F   DD  0B54                     PUSHL    #BUFSIZ
                 62   9F  0B5A                     PUSHAB   (R2)
        00000000'EF   9F  0B5C                     PUSHAB   A_PSLMNE
        00000000'EF   DD  0B62                     PUSHL    L_EXP_PSL
                 1F   DD  0B68                     PUSHL    #31
   000100B0 9F   0B   FB  0B6A                     CALLS    #11, a#DS$CVTREG
                          0B71   410           $DS_PRINTX_S     FORMAT=T_PSL,P0=L_EXP_PSL,P1=R2
                 52   DD  0B71                     PUSHL    R2
        00000000'EF   DD  0B73                     PUSHL    L_EXP_PSL
            FD8F CF   9F  0B79                     PUSHAB   T_PSL
   000100E8 9F   03   FB  0B7D                     CALLS    #$$N, a#DS$PRINTX
    5C F4B3 CF   0A   E1  0B84   411 14$:      BBC      #V_GPR_ERR,L_FLAGS,18$ ; br = no GPR error
                          0B8A   412           $DS_PRINTB_S     FORMAT=T_GPR
            FD9B CF   9F  0B8A                     PUSHAB   T_GPR
   000100E0 9F   01   FB  0B8E                     CALLS    #$$N, a#DS$PRINTB
   53   00000000'EF   DE  0B95   413           MOVAL    GPR_TABLE,R3                 ; load address of expected data
   54   00000000'EF   D0  0B9C   414           MOVL     A_HALT_BUFFER,R4             ; load address of actual data
                 55   7C  0BA3   415           CLRQ     R5                           ; init R5 and R6 for counter and data
            57   0D   D0  0BA5   416           MOVL     #13,R7                       ; load limit
58   00000000'EF45    DE  0BA8   417 15$:      MOVAL    REG_TABLE[R5],R8             ; load address of ascic string
   56   6445   6345   CD  0BB0   418           XORL3    (R3)[R5],(R4)[R5],R6         ; get XOR of data if bad
                 17   13  0BB6   419           BEQL     16$                          ; br = good data
                          0BB8   420           $DS_PRINTB_S     FORMAT=T_XOR,P0=R8,P1=(R3)[R5],P2=(R4)[R5],P3=R6
                 56   DD  0BB8                     PUSHL    R6
               6445   DD  0BBA                     PUSHL    (R4)[R5]
               6345   DD  0BBD                     PUSHL    (R3)[R5]
                 58   DD  0BC0                     PUSHL    R8
```

```
                   FD8B CF    9F   0BC2                        PUSHAB  T_XOR
          000100E0 9F    05   FB   0BC6                        CALLS   #$$N, a#DS$PRINTB
                         13   11   0BCD     421         BRB     17$                      ; check if finished
                                   0BCF     422 16$:    $DS_PRINTX_S    FORMAT=T_NOXOR,P0=R8,P1=(R3)[R5],P2=(R4)[R5]
                       6445   DD   0BCF                        PUSHL   (R4)[R5]
                       6345   DD   0BD2                        PUSHL   (R3)[R5]
                         58   DD   0BD5                        PUSHL   R8
                   FD90 CF    9F   0BD7                        PUSHAB  T_NOXOR
          000100E8 9F    04   FB   0BDB                        CALLS   #$$N, a#DS$PRINTX
                C2 55    57   F3   0BE2     423 17$:    AOBLEQ  R7,R5,15$
           0B F451 CF    05   E1   0BE6     424 18$:    BBC     #V_TIMER,L_FLAGS,19$     ; br = no timer error
                                   0BEC     425         $DS_PRINTB_S    FORMAT=T_TIMER
                   FAA1 CF    9F   0BEC                        PUSHAB  T_TIMER
          000100E0 9F    01   FB   0BF0                        CALLS   #$$N, a#DS$PRINTB
                                   0BF7     426 19$:
                                   0BF7     427         $DS_ENDMESSAGE
                              04   0BF7                        RET                      ; RETURN TO DIAGNOSTIC SUPERVISOR
                                   0BF8     428
```

```
                                              0BF8    430 .SBTTL  SUBROUTINES
                                              0BF8    431
                                              0BF8    432 ; This routine will clear the first 4 longwords of memory to disable
                                              0BF8    433 ; a Restart Parameter Block if one exists.
                                              0BF8    434
                                              0BF8    435 CLEAR_RPB:
                           54     00     D0   0BF8    436           MOVL     #0,R4             ; load R4 with starting address
                                  84     7C   0BFB    437           CLRQ     (R4)+             ; clear first two longwords
                                  64     7C   0BFD    438           CLRQ     (R4)              ; clear last two longwords
                                         05   0BFF    439           RSB                        ; return
                                              0C00    440
                                              0C00    441 ; This routine will setup a Restart Parameter Block at location 0
                                              0C00    442 ; that points to locaion 100 to handle processor halts when operating
                                              0C00    443 ; in automatic restart mode.
                                              0C00    444
                                              0C00    445 SETUP_RPB:
                           54     00     D0   0C00    446           MOVL     #0,R4             ; load R4 with address to build RPB
                           84     00     D0   0C03    447           MOVL     #0,(R4)+          ; load first longword with address
                  84   00000100'EF       DE   0C06    448           MOVAL    RESTART,(R4)+     ; load second longword with routine address
                  55   00000100'EF       DE   0C0D    449           MOVAL    RESTART,R5        ; load restart address to calc checksum
                                  56     D4   0C14    450           CLRL     R6                ; initialize R6 to store checksum
                       F3FD CF    01     D0   0C16    451           MOVL     #1,L_COUNT        ; initialize counter
                           56     85     C0   0C1B    452 1$:       ADDL2    (R5)+,R6          ; add longword checksum
                     F7 F3F5 CF   1F     F3   0C1E    453           AOBLEQ   #^X1F,L_COUNT,1$ ; br = not finished
                           84     56     D0   0C24    454           MOVL     R6,(R4)+          ; load RPB with checksum
                                  64     D4   0C27    455           CLRL     (R4)              ; disable warm restart flag
                                         05   0C29    456           RSB                        ; return
                                              0C2A    457
                                              0C2A    458 ; This routine will clear the buffer area.
                                              0C2A    459
                                              0C2A    460 CLEAR_BUFFER:
                  55   00000000'FF       DE   0C2A    461           MOVAL    @A_HALT_BUFFER,R5 ; get starting buffer address
                            F3E3 CF      D4   0C31    462           CLRL     L_COUNT           ; initialize counter
                                  85     D4   0C35    463 1$:       CLRL     (R5)+             ; clear a buffer location
         F4 F3D8 CF    00000100 8F       F3   0C37    464           AOBLEQ   #256,L_COUNT,1$ ; br = not finished
                                         05   0C41    465           RSB                        ; return
                                              0C42    466
                                              0C42    467 ; This routine will load general purpose registers R0 to R11 with data and
                                              0C42    468 ; read the AP and FP befor a halt is tested.
                                              0C42    469
                                              0C42    470 LOAD_GPR:
                  5B   00000000'EF       DE   0C42    471           MOVAL    GPR_TABLE,R11     ; load address of data table to load GPR's
                           50     8B     7D   0C49    472           MOVQ     (R11)+,R0         ; move data into R0 and R1
                           52     8B     7D   0C4C    473           MOVQ     (R11)+,R2         ; move data into R2 and R3
                           54     8B     7D   0C4F    474           MOVQ     (R11)+,R4         ; move data into R4 and R5
                           56     8B     7D   0C52    475           MOVQ     (R11)+,R6         ; move data into R6 and R7
                           58     8B     7D   0C55    476           MOVQ     (R11)+,R8         ; move data into R8 and R9
                           5A     8B     7D   0C58    477           MOVQ     (R11)+,R10        ; move data into R10 and R11
                  00000000'EF     5C     D0   0C5B    478           MOVL     AP,L_EXP_AP       ; move contents of AP into expected data table
                  00000000'EF     5D     D0   0C62    479           MOVL     FP,L_EXP_FP       ; move contents of FP into expected data table
                                         05   0C69    480           RSB                        ; return
                                              0C6A    481
                                              0C6A    482 ; This routine will read general purpose registers R0 to R11 , the AP, FP, SP
                                              0C6A    483 ; KSP, ISP and PSL into a buffer area (A_HALT_BUFFER) to check data patterns
                                              0C6A    484 ; and data after a halt. It will then retrun to to address contained in
```

```
                                            0C6A   485 ; RENTRY_POINT.
                                            0C6A   486
                                            0C6A   487 READ_GPR:
             00000000'FF     50     7D     0C6A   488         MOVQ    R0,@A_HALT_BUFFER ; move contents of R0 and R1 into buffer
             50    00000000'EF     D0     0C71   489         MOVL    A_HALT_BUFFER,R0 ; load R0 with address of buffer
                        50     08     C0     0C78   490         ADDL2   #8,R0               ; get over first two words
                        80     52     7D     0C7B   491         MOVQ    R2,(R0)+            ; move contents of R2 and R3 into buffer
                        80     54     7D     0C7E   492         MOVQ    R4,(R0)+            ; move R4 and R5
                        80     56     7D     0C81   493         MOVQ    R6,(R0)+            ; move R6 and R7
                        80     58     7D     0C84   494         MOVQ    R8,(R0)+            ; move R8 and R9
                        80     5A     7D     0C87   495         MOVQ    R10,(R0)+           ; move R10 and R11
                        80     5C     D0     0C8A   496         MOVL    AP,(R0)+            ; move AP
                        80     5D     D0     0C8D   497         MOVL    FP,(R0)+            ; move FP
             00000000'EF     5E     D0     0C90   498         MOVL    SP,L_ACT_SP         ; move SP
   00000000'EF    00000000'8F     DB     0C97   499         MFPR    #PR$_KSP,L_ACT_KSP ; move KSP
   00000000'EF    00000000'8F     DB     0CA2   500         MFPR    #PR$_ISP,L_ACT_ISP ; move ISP
                  00000000'EF     DC     0CAD   501         MOVPSL  L_ACT_PSL           ; move PSL
                     F34D DF     17     0CB3   502         JMP     @RENTRY_POINT       ; return to subtest
                                            0CB7   503
                                            0CB7   504
                                            0CB7   505 ; This routine will service unexpected interval timer interrupts and set an
                                            0CB7   506 ; error flag.
                                            0CB7   507
                                            0CB7   508         .ALIGN  LONG
                                            0CB8   509 TIMER_SERVICE:
                        00'    00     DA     0CB8   510         MTPR    #0,S^#PR$_ICCS              ; Turn off the timer
                   F37C CF     20     C8     0CBB   511         BISL2   #M_TIMER,L_FLAGS            ; set error flag
                                      02     0CC0   512         REI                                 ; return
                                            0CC1   513
                                            0CC1   514
                                            0CC1   515 ; This routine will service unexpected CHANGE MODE exceptions and set
                                            0CC1   516 ; an error flag.
                                            0CC1   517
                                            0CC1   518         .ALIGN  LONG
                                            0CC4   519 CHMX_SERVICE:
                   F373 CF     08     C8     0CC4   520         BISL2   #M_CHMX_ERR,L_FLAGS ; set error flag
                               8E     D5     0CC9   521         TSTL    (SP)+               ; get change mode argument off stack
                                      02     0CCB   522         REI                         ; return
                                            0CCC   523
                                            0CCC   524 ; This routine will service unexpected XFC exceptions and set a flag
                                            0CCC   525 ; to indicate the error.
                                            0CCC   526
                                            0CCC   527         .ALIGN  LONG
                                            0CCC   528 XFC_ERROR:
                   F36B CF     10     C8     0CCC   529         BISL2   #M_XFC_ERR,L_FLAGS ; set error flag
                        6E     01     C0     0CD1   530         ADDL2   #1,(SP)             ; update the pc for the REI
                                      02     0CD4   531         REI                         ; return
                                            0CD5   532
                                            0CD5   533 ; This routine will service expected XFC exceptions and set a flag to
                                            0CD5   534 ; indicate successful completion of the operation.
                                            0CD5   535
                                            0CD5   536         .ALIGN  LONG
                                            0CD8   537 XFC_SERVICE:
      F35B CF    00000800 8F     C8     0CD8   538         BISL2   #M_XFC,L_FLAGS      ; set flag
                        6E     01     C0     0CE1   539         ADDL2   #1,(SP)             ; update the pc for the REI
```

```
                                   02  0CE4    540            REI                          ; return
                                       0CE5    541
                                       0CE5    542 ; This routine is called after execution of an instruction which should cause
                                       0CE5    543 ; a halt. The PSL, SP, KSP, ISP, GENERAL PURPOSE REGISTERS and FLAG BYTE are
                                       0CE5    544 ; examined for expected information. If an error exists, a flag is set to
                                       0CE5    545 ; indicate that it should be reported.
                                       0CE5    546
                                       0CE5    547 ERROR_CHECK::
                             57    D4  0CE5    548            CLRL     R7                  ; initialize counter
                       55    0D    D0  0CE7    549            MOVL     #13,R5              ; set limit
              53   00000000'EF     D0  0CEA    550            MOVL     A_HALT_BUFFER,R3 ; load address of actual data to check
              54   00000000'EF     DE  0CF1    551            MOVAL    GPR_TABLE,R4        ; load address of expected data
                       84    83    D1  0CF8    552 1$:        CMPL     (R3)+,(R4)+         ; check data
                             09    13  0CFB    553            BEQL     2$                  ; br = data is good
        F336 CF    00001400 8F     C8  0CFD    554            BISL2    #M_ERROR_FLAG!M_GPR_ERR,L_FLAGS ; set GPR and error flag
                    EE 57    55    F3  0D06    555 2$:        AOBLEQ   R5,R7,1$            ; br = not finished
   00000000'EF     00000000'EF     D1  0D0A    556            CMPL     L_ACT_SP,L_EXP_SP ; check SP
                             09    13  0D15    557            BEQL     3$                  ; br = SP is good
        F31C CF    00009000 8F     C8  0D17    558            BISL2    #M_SP_ERR!M_ERROR_FLAG,L_FLAGS ; set SP and error flag
          1F 00000000'EF     1A    E1  0D20    559 3$:        BBC      #PSL$V_IS,L_ACT_PSL,4$ ; br = on KS no possible error
        F30B CF    00010000 8F     C8  0D28    560            BISL2    #M_ISTK,L_FLAGS ; set interrupt stack flag
   00000000'EF     00000000'EF     D1  0D31    561            CMPL     L_ACT_KSP,L_EXP_KSP ; check KSP
                             09    13  0D3C    562            BEQL     4$                  ; br = KSP is good
        F2F5 CF    00003000 8F     C8  0D3E    563            BISL2    #M_KSP_ERR!M_ERROR_FLAG,L_FLAGS ; set KSP and error flag
             16 F2F0 CF      10    E0  0D47    564 4$:        BBS      #V_ISTK,L_FLAGS,5$ ; br = on IS no possible error
   00000000'EF     00000000'EF     D1  0D4D    565            CMPL     L_ACT_ISP,L_EXP_ISP ; check ISP
                             09    13  0D58    566            BEQL     5$                  ; br = ISP is good
        F2D9 CF    00005000 8F     C8  0D5A    567            BISL2    #M_ISP_ERR!M_ERROR_FLAG,L_FLAGS ; set ISP and error flag
   00000000'EF     00000000'EF     D1  0D63    568 5$:        CMPL     L_ACT_PSL,L_EXP_PSL ; check PSL
                             42    13  0D6E    569            BEQL     8$                  ; br = PSL is good
                       02    18    EF  0D70    570            EXTZV    #PSL$V_CURMOD,#PSL$S_CURMOD,-
        F2A9 CF    00000000'EF         0D73    571                     L_EXP_PSL,L_EXP_MOD ; extract current mode from expected psl
                       02    18    ED  0D7B    572            CMPZV    #PSL$V_CURMOD,#PSL$S_CURMOD,-
        F29E CF    00000000'EF         0D7E    573                     L_ACT_PSL,L_EXP_MOD ; check current mode for error
                             18    12  0D86    574            BNEQ     6$                  ; br = wrong mode
                       01    1A    EF  0D88    575            EXTZV    #PSL$V_IS,#1,-
        F295 CF    00000000'EF         0D8B    576                     L_EXP_PSL,L_EXP_STACK ; extract stack from expected PSL
                       01    1A    ED  0D93    577            CMPZV    #PSL$V_IS,#1,-
        F28A CF    00000000'EF         0D96    578                     L_ACT_PSL,L_EXP_STACK ; check current stack for error
                             09    13  0D9E    579            BEQL     7$                  ; br = mode is good
        F293 CF    00000100 8F     C8  0DA0    580 6$:        BISL2    #M_FATAL_ERR,L_FLAGS ; signal fatal error
        F28A CF    00001200 8F     C8  0DA9    581 7$:        BISL2    #M_PSL_ERR!M_ERROR_FLAG,L_FLAGS ; set PSL and error flag
             09 F285 CF      00    E0  0DB2    582 8$:        BBS      #V_READY,L_FLAGS,9$ ; br = no ready error
        F27B CF    00001000 8F     C8  0DB8    583            BISL2    #M_ERROR_FLAG,L_FLAGS ; set error flag
             09 F276 CF      02    E1  0DC1    584 9$:        BBC      #V_EXECUTE_ERR,L_FLAGS,10$ ; br = no execute error
        F26C CF    00001000 8F     C8  0DC7    585            BISL2    #M_ERROR_FLAG,L_FLAGS ; set error flag
             09 F267 CF      01    E0  0DD0    586 10$:       BBS      #V_DONE,L_FLAGS,11$ ; br = no done error
        F25D CF    00001000 8F     C8  0DD6    587            BISL2    #M_ERROR_FLAG,L_FLAGS ; set error flag
             09 F258 CF      03    E1  0DDF    588 11$:       BBC      #V_CHMX_ERR,L_FLAGS,12$ ; br = no unexpected CHMx exception
        F24E CF    00001000 8F     C8  0DE5    589            BISL2    #M_ERROR_FLAG,L_FLAGS ; set error flag
             09 F249 CF      04    E1  0DEE    590 12$:       BBC      #V_XFC_ERR,L_FLAGS,13$ ; br = no unexpected XFC exception
        F23F CF    00001000 8F     C8  0DF4    591            BISL2    #M_ERROR_FLAG,L_FLAGS ; set error flag
             09 F23A CF      05    E1  0DFD    592 13$:       BBC      #V_TIMER,L_FLAGS,14$ ; br = no unexpected timer interrupt
        F230 CF    00001000 8F     C8  0E03    593            BISL2    #M_ERROR_FLAG,L_FLAGS ; set error flag
                                   05  0E0C    594 14$:       RSB                          ; return
```

```
    0E0D   595
    0E0D   596         $DS_PAGE
```

```
                                     0E0D               .SBTTL  TEST 4:  PROCESSOR HALTS ARCHITECTURALLY DEFINED
                                 00000000                       .PSECT  TEST_004, PAGE, NOWRT
                                     002C
                                     002C   599 ;++
                                     002C   600 ; Test description:
                                     002C   601 ;
                                     002C   602 ;       This test checks the various ways the VAX-11/750 processor can
                                     002C   603 ;       be halted in an architecturally defined manner. This includes
                                     002C   604 ;       executing a HALT instruction in kernel mode, attempting
                                     002C   605 ;       certain references to an invalid interrupt stack and executing
                                     002C   606 ;       change mode instructions when the IS bit of the PSL is set.
                                     002C   607 ;
                                     002C   608 ;--
                                     002C   609
                                     002C   610
                                     002C   611         $DS_BGNTEST     <MANUAL,HALT,ALL>
                                     002C       DATA_004:
                           00000000  002C                       .LONG   0               ; TEST ARGUMENT TABLE TERMINATOR
                                     0030       TEST_004::
                               0000  0030                       .WORD   ^M<>            ; ENTRY MASK
                                     0032   612         $DS_BOPER  5$                   ; breach if operatoer present
                            0C    E0  0032                      BBS     #DSA$V_OPER,-   ; BR IF OPERATOR
                14 0000FE00 9F        0034                              @#DSA$GL_FLAGS, 5$
                                     003A   613         $DS_PRINTF_S    FORMAT=T_NO_OPER        ;5.2
                   0000004D'EF   9F  003A                       PUSHAB  T_NO_OPER
              000100F0 9F   01   FB  0040                       CALLS   #$$N, @#DS$PRINTF
                                     0047   614         $DS_ABORT                       ; else, abort
              00010020 9F   6C   FA  0047                       CALLG   (AP), @#DS$ABORT
                                     004E   615 5$:     $DS_PRINTF_S    FORMAT=T_WARNING
                   00000262'EF   9F  004E                       PUSHAB  T_WARNING
              000100F0 9F   01   FB  0054                       CALLS   #$$N, @#DS$PRINTF
              00000000'EF   5E   D0  005B   616         MOVL    SP,L_KSP                ; save initial KSP
     00000000'EF   00000000'8F   DB  0062   617         MFPR    #PR$_ISP,L_ISP          ; save initial ISP
                   00000C2A'EF   16  006D   618         JSB     CLEAR_BUFFER            ; clear buffer area
                                     0073   619         $DS_SETVEC_S    VECTOR=#SCB$L_TIMER, SRVADR=TIMER_SERVICE
                            00   DD  0073                       PUSHL   #0
                   00000CB8'EF   DF  0075                       PUSHAL  TIMER_SERVICE
                   000000C0 8F   DD  007B                       PUSHL   #SCB$L_TIMER
              00010160 9F   03   FB  0081                       CALLS   #3, @#DS$SETVEC
                                     0088   620 1$:
                                     0088   621         $DS_PAGE
```

```
                                0088              .SBTTL  SUBTEST 4.1:  HALT instruction in Kernel Mode
                                0088   624 ;+
                                0088   625 ; Subtest description:
                                0088   626 ;
                                0088   627 ;      This subtest checks that the HALT instruction halts the processor
                                0088   628 ;      when executed in kernel mode (code = 6).
                                0088   629 ;
                                0088   630 ; Assumptions:
                                0088   631 ;
                                0088   632 ; Subtest steps:
                                0088   633 ;
                                0088   634 ;      SUBTEST Halt instruction in Kernel Mode
                                0088   635 ;      :  Clear RPB from location 0 in memory
                                0088   636 ;      :  Initialize Flags
                                0088   637 ;      :  Setup code to set done flag at loc 100
                                0088   638 ;      :  Setup a jump to READ_GPR point at next loc
                                0088   639 ;      :  IF operator present
                                0088   640 ;      :  :  THEN
                                0088   641 ;      :  :  Print instructions to operator
                                0088   642 ;      :  ENDIF
                                0088   643 ;      :  IF operator NOT present
                                0088   644 ;      :  :  THEN
                                0088   645 ;      :  :  Setup RPB at loc 0 pointing to loc 100
                                0088   646 ;      :  ENDIF
                                0088   647 ;      :  Load expected data for stacks
                                0088   648 ;      :  Load GPR's with data
                                0088   649 ;      :  Set ready to execute flag
                                0088   650 ;      :  Execute HALT
                                0088   651 ;      :  IF halt not executed properly
                                0088   652 ;      :  :  THEN
                                0088   653 ;      :  :  (will continue inline execution and get here)
                                0088   654 ;      :  :  Set halt execution error flag
                                0088   655 ;      :  ENDIF
                                0088   656 ;      :  Rentry point (restart will enter here)
                                0088   657 ;      :  Go check FLAGS, PSL, KSP, ISP, SP and GPR's for errors
                                0088   658 ;      :  IF any errors
                                0088   659 ;      :  :  THEN
                                0088   660 ;      :  :  Report errors
                                0088   661 ;      :  ENDIF
                                0088   662 ;      :  IF fatal error
                                0088   663 ;      :  :  THEN
                                0088   664 ;      :  :  Abort test
                                0088   665 ;      :  ENDIF
                                0088   666 ;      ENDSUBTEST Halt instruction in kernel mode
                                0088   667 ;
                                0088   668 ;-
                                0088   669
                                0088   670        $DS_BGNSUB
                                0088       T4_S1::
00010030 9F   00000000'EF   FA  0088                  CALLG   $S$, @#DS$BGNSUB
              00000BF8'EF   16  0093   671        JSB     CLEAR_RPB               ; disable restart parameter block
              00000C2A'EF   16  0099   672        JSB     CLEAR_BUFFER            ; clear data buffers
              0000003C'EF   D4  009F   673        CLRL    L_FLAGS                 ; initialize flags
00000100'EF   00000040'EF   7D  00A5   674        MOVQ    JUMP,RESTART            ; load loc 100 with rentry code
00000108'EF   00000048'EF   7D  00B0   675        MOVQ    JUMP+8,RESTART+8        ; load rest of rentry code
```

```
     00000004'EF   00000159'EF   DE  00BB   676         MOVAL   RENTRY1,RENTRY_POINT    ; load rentry address
           08 0000FE00'EF   0C   E0  00C6   677         BBS     #DSA$V_OPER,DSA$GL_FLAGS,1$ ; skip RPB if operator present
                   00000C00'EF   16  00CE   678         JSB     SETUP_RPB               ; build restart parameter block
                            22   11  00D4   679         BRB     2$                      ; skip message to operator
                                     00D6   680 1$:     $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS, P0=#4$, P1=#CODE_06
                            06   DD  00D6                       PUSHL   #CODE_06
                   00000152'8F   DD  00D8                       PUSHL   #4$
                   000001DC'EF   9F  00DE                       PUSHAB  T_INSTRUCTIONS
              000100F0 9F   03   FB  00E4                       CALLS   #$$N, @#DS$PRINTF
                                     00EB   681         $DS_PRINTF_S    FORMAT=T_CONTINUE
                   00000212'EF   9F  00EB                       PUSHAB  T_CONTINUE
              000100F0 9F   01   FB  00F1                       CALLS   #$$N, @#DS$PRINTF
     00000000'EF   00000000'EF   D0  00F8   682 2$:     MOVL    L_KSP,L_EXP_SP          ; load expected SP
                   00000000'EF   D4  0103   683         CLRL    L_EXP_KSP               ; KSP unpredictable if on kernel stack
     00000000'EF   00000000'EF   D0  0109   684         MOVL    L_ISP,L_EXP_ISP         ; load expected ISP
     00000000'EF   0000097F'EF   D0  0114   685         MOVL    DATA_1,L_EXP_PSL        ; load expected PSL
                   00000C42'EF   16  011F   686         JSB     LOAD_GPR                ; load GPR data and read AP & FP
           1D 0000FE00'EF   0C   E0  0125   687         BBS     #DSA$V_OPER,DSA$GL_FLAGS,3$ ; br = operator present
     00000000'EF   00000983'EF   D0  012D   688         MOVL    DATA_1+4,L_EXP_PSL      ; load PSL for auto restart
              00000000'EF   06   D0  0138   689         MOVL    #6,L_EXP_AP             ; load AP for auto restart
     00000000'EF   00000200 8F   D0  013F   690         MOVL    #^X200,L_EXP_SP         ; load SP for auto restart
              0000003C'EF   01   C8  014A   691 3$:     BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
                            00   00  0151   692         HALT                            ; execute HALT instruction (should
                                     0152   693                                         ; halt and continue at rentry point)
              0000003C'EF   04   C8  0152   694 4$:     BISL2   #M_EXECUTE_ERR,L_FLAGS  ; halt not executed, set error flag
                                     0159   695 RENTRY1:                                ; rentry point after executing halt
              5E   00000000'EF   D0  0159   696         MOVL    L_KSP,SP                ; reset SP
                   00000CE5'EF   16  0160   697         JSB     ERROR_CHECK             ; check for errors
           23 0000003C'EF   0C   E1  0166   698         BBC     #V_ERROR_FLAG,L_FLAGS,5$ ; br = no errors
                                     016E   699         $DS_ERRHARD_S   UNIT =#0,PRLINK=ERR_MESSAGE,P1=#T_HALT_ERR
                   000002F1'8F   DD  016E                       PUSHL   #T_HALT_ERR
                   0000099B'EF   DF  0174                       PUSHAL  ERR_MESSAGE
                            00   DD  017A                       PUSHL   #0
                            00   DD  017C                       PUSHL   #0
                            01   DD  017E                       PUSHL   #$ER
                                     0180       ;;; TEST 4, SUBTEST 1, ERROR 1
              000100D0 9F   05   FB  0180                       CALLS   #$$M, @#DS$ERRHARD
                                     0187   700         $DS_BCOMPLETE   5$
                         07 50   E8  0187                       BLBS    R0, 5$
                                     018A   701         $DS_ABORT
              00010020 9F   6C   FA  018A                       CALLG   (AP), @#DS$ABORT
           07 0000003C'EF   08   E1  0191   702 5$:     BBC     #V_FATAL_ERR,L_FLAGS,6$ ; br = no fatal errors
                                     0199   703         $DS_ABORT
              00010020 9F   6C   FA  0199                       CALLG   (AP), @#DS$ABORT
                                     01A0   704 6$:     $DS_ENDSUB
                                     01A0       T4_S1_X:
     00010038 9F   00000000'EF   FA  01A0                       CALLG   $S$, @#DS$ENDSUB
                                     01AB   705         $DS_PAGE
```

```
01AB                 .SBTTL  SUBTEST 4.2:  Interrupt Stack invalid
01AB       ;+
01AB       ; Subtest description:
01AB       ;
01AB       ;       This subtest checks that an interrupt stack which is located on
01AB       ;       a page which is protected to kernel read only, will halt the processor
01AB       ;       when a write is attempted. An XFC instruction is executed to get onto
01AB       ;       the interrupt stack prior to invalidating the interrupt stack pointer.
01AB       ;       A second XFC instruction is executed with the vector specifying the
01AB       ;       interrupt stack for service. When the XFC instruction is executed
01AB       ;       there is an attempt to write the PC and PSL on the interrupt stack.
01AB       ;       When this happens an "Interrupt Stack not valid HALT  should occur
01AB       ;       (halt code = 4). Memory management is on for this subtest.
01AB       ;
01AB       ; Subtest steps:
01AB       ;
01AB       ;       SUBTEST Interrupt Stack invalid
01AB       ;       :  Initialize Flags
01AB       ;       :  Clear RPB from location 0 in memory
01AB       ;       :  Turn memory management on
01AB       ;       :  Set page0 protection to kernel write access
01AB       ;       :  IF unsucessful
01AB       ;       :  :  THEN
01AB       ;       :  :  Report system error
01AB       ;       :  :  Abort program
01AB       ;       :  ENDIF
01AB       ;       :  Get XFC vector to handle on interrupt stack
01AB       ;       :  Execute XFC instruction
01AB       ;       :  IF not on the interrupt stack
01AB       ;       :  :  THEN
01AB       ;       :  :  Report system error
01AB       ;       :  :  Abort program
01AB       ;       :  ENDIF
01AB       ;       :  Release XFC vector
01AB       ;       :  Get XFC vector to handle on interrupt stack
01AB       ;       :  Setup code to set done flag at loc 100
01AB       ;       :  Setup a jump to READ_GPR at next loc
01AB       ;       :  Save current IS SP to reset after subtest is complete
01AB       ;       :  Set interrupt stack pointer to buffer area
01AB       ;       :  Invalidate buffer page (kernel read only access)
01AB       ;       :  IF unsucessful
01AB       ;       :  :  THEN
01AB       ;       :  :  Report system error
01AB       ;       :  :  Abort program
01AB       ;       :  ENDIF
01AB       ;       :  IF operator present
01AB       ;       :  :  THEN
01AB       ;       :  :  Print instructions to operator
01AB       ;       :  ENDIF
01AB       ;       :  IF operator NOT present
01AB       ;       :  :  THEN
01AB       ;       :  :  Setup RPB at loc 0 pointing to loc 100
01AB       ;       :  ENDIF
01AB       ;       :  Setup all expected data
01AB       ;       :  Load GPR's with data
```

```
                                     01AB    762 ;        :  Set ready to execute flag
                                     01AB    763 ;        :  Execute XFC to cause exception to attempt to write PC and PSL on IS
                                     01AB    764 ;        :  (this should cause an interrupt stack not valid HALT)
                                     01AB    765 ;        :  IF XFC not executed properly
                                     01AB    766 ;        :  :  THEN
                                     01AB    767 ;        :  :  (will continue in line execution and get here)
                                     01AB    768 ;        :  :  Set XFC execution error flag
                                     01AB    769 ;        :  ENDIF
                                     01AB    770 ;        :  Rentry point (restart will enter here)
                                     01AB    771 ;        :  Read  PSL
                                     01AB    772 ;        :  IF on IS as expected
                                     01AB    773 ;        :  :  THEN
                                     01AB    774 ;        :  :  Restore IS SP
                                     01AB    775 ;        :  ENDIF
                                     01AB    776 ;        :  Go check FLAGS, PSL, KSP, ISP, SP and GPR'S for errors
                                     01AB    777 ;        :   (Either the KSP or ISP will be UNPREDICTABLE depending on the
                                     01AB    778 ;        :   current stack, this is machine dependant)
                                     01AB    779 ;        :  IF any errors
                                     01AB    780 ;        :  :  THEN
                                     01AB    781 ;        :  :  Report errors
                                     01AB    782 ;        :  ENDIF
                                     01AB    783 ;        :  IF fatal error
                                     01AB    784 ;        :  :  THEN
                                     01AB    785 ;        :  :  Abort program
                                     01AB    786 ;        :  ENDIF
                                     01AB    787 ;        :  Reset protection on buffer page to original protection
                                     01AB    788 ;        :  Turn memory management off
                                     01AB    789 ;        :  Release XFC vector
                                     01AB    790 ;        ENDSUBTEST Interrupt Stack invalid
                                     01AB    791 ;
                                     01AB    792 ;-
                                     01AB    793
                                     01AB    794          $DS_BGNSUB
                                     01AB         T4_S2::
      00010030 9F   0000000C'EF   FA  01AB                       CALLG   $S$, @#DS$BGNSUB
                    00000BF8'EF   16  01B6    795          JSB     CLEAR_RPB                ; disable restart parameter block
                    00000C2A'EF   16  01BC    796          JSB     CLEAR_BUFFER             ; clear data buffers
                    0000003C'EF   D4  01C2    797          CLRL    L_FLAGS                  ; initialize flags
                    00000010'EF   D4  01C8    798          CLRL    L_PAGEZERO               ; set page zero to 0
                                     01CE    799          $DS_MMON_S                       ; turn memory management on
             00010150 9F     00   FB  01CE                       CALLS   #0, @#DS$MMON
                                     01D5    800          $SETPRT_S        INADR=L_PAGEZERO,PROT=#PRT$C_KW
                             00   DD  01D5                       PUSHL   #0
                    00000000'8F   DD  01D7                       PUSHL   #PRT$C_KW
                             00   DD  01DD                       PUSHL   #0
                             00   DD  01DF                       PUSHL   #0
                    00000010'EF   7F  01E1                       PUSHAQ  L_PAGEZERO
             00010430'GF     05   FB  01E7                       CALLS   #5,G^SYS$SETPRT
                                     01EE    801          $DS_BCOMPLETE    1$
                          1A 50   E8  01EE                       BLBS    R0, 1$
                                     01F1    802          $DS_ERRSYS_S     UNIT=#0,MSGADR=T_PAGE0PROT
                             00   DD  01F1                       PUSHL   #0
                    00000816'EF   DF  01F3                       PUSHAL  T_PAGE0PROT
                             00   DD  01F9                       PUSHL   #0
                             01   DD  01FB                       PUSHL   #$ER
```

```
                                     01FD       ;;; TEST 4, SUBTEST 2, ERROR 1
              000100C0 9F   04   FB  01FD                       CALLS   #$$M, @#DS$ERRSYS
                                     0204   803         $DS_ABORT
              00010020 9F   6C   FA  0204                       CALLG   (AP), @#DS$ABORT
                                     020B   804 1$:     $DS_SETVEC_S    VECTOR=#SCB$L_OPCCUS,SRVADR=RENTRY2,CODE=#1
                            01   DD  020B                       PUSHL   #1
                   00000220'EF   DF  020D                       PUSHAL  RENTRY2
                            14   DD  0213                       PUSHL   #SCB$L_OPCCUS
              00010160 9F   03   FB  0215                       CALLS   #3, @#DS$SETVEC
                                 FC  021C   805 2$:     XFC
                                     021D   806         .ALIGN  LONG
                                     0220   807 RENTRY2:
                   00000000'EF   DC  0220   808         MOVPSL  L_ACT_PSL
           1A 00000000'EF   1A   E0  0226   809         BBS     #PSL$V_IS,L_ACT_PSL,3$
                                     022E   810         $DS_ERRHARD_S   UNIT=#0,MSGADR=T_NOT_ON_IS
                            00   DD  022E                       PUSHL   #0
                   0000089B'EF   DF  0230                       PUSHAL  T_NOT_ON_IS
                            00   DD  0236                       PUSHL   #0
                            02   DD  0238                       PUSHL   #$ER
                                     023A       ;;; TEST 4, SUBTEST 2, ERROR 2
              000100D0 9F   04   FB  023A                       CALLS   #$$M, @#DS$ERRHARD
                                     0241   811         $DS_ABORT
              00010020 9F   6C   FA  0241                       CALLG   (AP), @#DS$ABORT
                                     0248   812 3$:     $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                            14   DD  0248                       PUSHL   #SCB$L_OPCCUS
              00010168 9F   01   FB  024A                       CALLS   #1, @#DS$CLRVEC
                                     0251   813         $DS_SETVEC_S    VECTOR=#SCB$L_OPCCUS,SRVADR=XFC_SERVICE,CODE=#1
                            01   DD  0251                       PUSHL   #1
                   00000CD8'EF   DF  0253                       PUSHAL  XFC_SERVICE
                            14   DD  0259                       PUSHL   #SCB$L_OPCCUS
              00010160 9F   03   FB  025B                       CALLS   #3, @#DS$SETVEC
     00000100'EF   00000040'EF   7D  0262   814 4$:     MOVQ    JUMP,RESTART            ; setup code to renter after halt
     00000108'EF   00000048'EF   7D  026D   815         MOVQ    JUMP+8,RESTART+8        ; load rest of rentry code
     00000004'EF   00000363'EF   DE  0278   816         MOVAL   RENTRY3,RENTRY_POINT    ; load rentry address
           08 0000FE00'EF   0C   E0  028A   818         BBS     #DSA$V_OPER,DSA$GL_FLAGS,5$ ; br = operator present
                   00000C00'EF   16  0292   819         JSB     SETUP_RPB               ; no operator, use auto restart
                            22   11  0298   820         BRB     6$                      ; skip over message
                                     029A   821 5$:     $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#8$,P1=#CODE_04
                            04   DD  029A                       PUSHL   #CODE_04
                   0000035B'8F   DD  029C                       PUSHL   #8$
                   000001DC'EF   9F  02A2                       PUSHAB  T_INSTRUCTIONS
              000100F0 9F   03   FB  02A8                       CALLS   #$$N, @#DS$PRINTF
                                     02AF   822         $DS_PRINTF_S    FORMAT=T_CONTINUE
                   00000212'EF   9F  02AF                       PUSHAB  T_CONTINUE
              000100F0 9F   01   FB  02B5                       CALLS   #$$N, @#DS$PRINTF
     0000001C'EF   00000000'EF   D0  02BC   823 6$:     MOVL    A_BUFFER_BEGIN,L_ADDRESS ; get starting address of buffer
     00000008'EF   0000001C'EF   D0  02C7   824         MOVL    L_ADDRESS,L_PAGESTRT    ; load address of page to change
     0000000C'EF   0000001C'EF   D0  02D2   825         MOVL    L_ADDRESS,L_PAGEND      ; protection in 2 longword array
              0000001C'EF   28   C0  02DD   826         ADDL2   #40,L_ADDRESS           ; point into protected page
                                     02E4   827         $SETPRT_S INADR=L_PAGESTRT,PROT-#PRT$C_KR,PRVPRT=L_OLDPROT
                   0000002C'EF   9F  02E4                       PUSHAB  L_OLDPROT
                   00000000'8F   DD  02EA                       PUSHL   #PRT$C_KR
                            00   DD  02F0                       PUSHL   #0
                            00   DD  02F2                       PUSHL   #0
```

```
                     00000008'EF    7F   02F4                         PUSHAQ  L_PAGESTRT
              00010430'GF     05    FB   02FA                         CALLS   #5,G^SYS$SETPRT
                                         0301    828         $DS_BCOMPLETE    7$
                        1A 50       E8   0301                         BLBS    R0, 7$
                                         0304    829         $DS_ERRSYS_S     UNIT=#0,MSGADR=T_PAGEPROT
                              00    DD   0304                         PUSHL   #0
                     00000844'EF    DF   0306                         PUSHAL  T_PAGEPROT
                              00    DD   030C                         PUSHL   #0
                              03    DD   030E                         PUSHL   #$ER
                                         0310        ;;; TEST 4, SUBTEST 2, ERROR 3
              000100C0 9F     04    FB   0310                         CALLS   #$$M, @#DS$ERRSYS
                                         0317    830         $DS_ABORT
              00010020 9F     6C    FA   0317                         CALLG   (AP), @#DS$ABORT
                                         031E    831
00000000'EF   0000001C'EF     04    C3   031E    832 7$:     SUBL3   #4,L_ADDRESS,L_EXP_SP    ; load expected SP
     00000000'EF    00000000'EF     D0   032A    833         MOVL    L_KSP,L_EXP_KSP          ; load expected KSP
                    00000000'EF     D4   0335    834         CLRL    L_EXP_ISP                ; ISP unpredictable if on int. stack
     00000000'EF    00000983'EF     D0   033B    835         MOVL    DATA_2,L_EXP_PSL         ; load expected PSL
                                         0346    836
                    00000C42'EF     16   0346    837         JSB     LOAD_GPR                 ; load GPR's with data and read AP, FP
              0000003C'EF     01    C8   034C    838         BISL2   #M_READY,L_FLAGS         ; set ready to execute flag
              5E    0000001C'EF     D0   0353    839         MOVL    L_ADDRESS,SP             ; point IS SP to that address
                              01         035A    840         NOP                              ; wait a little
                                    FC   035B    841 8$:     XFC                              ; execute XFC instruction to cause
                                         035C    842                                          ; exception which will attempt to push
                                         035C    843                                          ; on an invalid IS, renter at RENTRY2
                                         035C    844
              0000003C'EF     04    C8   035C    845         BISL2   #M_EXECUTE_ERR,L_FLAGS   ; set flag, XFC did not work properly
                                         0363    846
                                         0363    847 RENTRY3:
                                         0363    848
           11 00000000'EF     1A    E1   0363    849         BBC     #PSL$V_IS,L_ACT_PSL,10$ ; br = not on IS as expected
              5E    00000020'EF     D0   036B    850         MOVL    L_CUR_ISP,SP             ; reset IS SP
              6E    0000037A'8F     D0   0372    851         MOVL    #9$,(SP)                 ; setup return address
                              02         0379    852         REI                              ; get off interrupt stack
     00000000'8F    00000020'EF     DA   037C    854 10$:    MTPR    L_CUR_ISP,#PR$_ISP       ; load IS SP into IPR
                    00000CE5'EF     16   0387    855 11$:    JSB     ERROR_CHECK              ; check for errors (GPR's,AP,FP,PSL)
           20 0000003C'EF     0C    E1   038D    856         BBC     #V_ERROR_FLAG,L_FLAGS,12$ ; br = no errors
                                         0395    857         $DS_ERRHARD_S    UNIT=#0,PRLINK=ERR_MESSAGE,P1=#T_INVLDIS_ERR
                    00000315'8F     DD   0395                         PUSHL   #T_INVLDIS_ERR
                    0000099B'EF     DF   039B                         PUSHAL  ERR_MESSAGE
                              00    DD   03A1                         PUSHL   #0
                              00    DD   03A3                         PUSHL   #0
                              04    DD   03A5                         PUSHL   #$ER
                                         03A7        ;;; TEST 4, SUBTEST 2, ERROR 4
              000100D0 9F     05    FB   03A7                         CALLS   #$$M, @#DS$ERRHARD
                                         03AE    858         $DS_ABORT
              00010020 9F     6C    FA   03AE                         CALLG   (AP), @#DS$ABORT
           07 0000003C'EF     08    E1   03B5    859 12$:    BBC     #V_FATAL_ERR,L_FLAGS,13$ ; br = no fatal errors
                                         03BD    860         $DS_ABORT
              00010020 9F     6C    FA   03BD                         CALLG   (AP), @#DS$ABORT
                                         03C4    861 13$:    $SETPRT_S INADR=L_PAGESTRT,PROT=#PRT$C_KW,PRVPRT=L_OLDPROT
                    0000002C'EF     9F   03C4                         PUSHAB  L_OLDPROT
                    00000000'8F     DD   03CA                         PUSHL   #PRT$C_KW
```

```
                            00   DD  03D0                           PUSHL   #0
                            00   DD  03D2                           PUSHL   #0
                   00000008'EF   7F  03D4                           PUSHAQ  L_PAGESTRT
              00010430'GF   05   FB  03DA                           CALLS   #5,G^SYS$SETPRT
                                     03E1   862         $DS_BCOMPLETE   14$
                         1A 50   E8  03E1                           BLBS    R0, 14$
                                     03E4   863         $DS_ERRSYS_S    UNIT=#0,MSGADR=T_PAGEPROT
                            00   DD  03E4                           PUSHL   #0
                   00000844'EF   DF  03E6                           PUSHAL  T_PAGEPROT
                            00   DD  03EC                           PUSHL   #0
                            05   DD  03EE                           PUSHL   #$ER
                                     03F0       ;;; TEST 4, SUBTEST 2, ERROR 5
              000100C0 9F   04   FB  03F0                           CALLS   #$$M, @#DS$ERRSYS
                                     03F7   864         $DS_ABORT
              00010020 9F   6C   FA  03F7                           CALLG   (AP), @#DS$ABORT
                                     03FE   865 14$:    $DS_MMOFF_S                     ; turn memory management off
              00010158 9F   00   FB  03FE                           CALLS   #0, @#DS$MMOFF
                                     0405   866         $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                            14   DD  0405                           PUSHL   #SCB$L_OPCCUS
              00010168 9F   01   FB  0407                           CALLS   #1, @#DS$CLRVEC
                                     040E   867         $DS_ENDSUB
                                     040E       T4_S2_X:
     00010038 9F   0000000C'EF   FA  040E                           CALLG   $S$, @#DS$ENDSUB
                                     0419   868         $DS_PAGE
```

```
0419              .SBTTL  SUBTEST 4.3:  IS SP points to NXM
0419      ;+
0419      ; Subtest description:
0419      ;
0419      ;       This subtest checks that an interrupt stack, stack pointer which
0419      ;       points to nonexistent memory will halt the processor. Memory
0419      ;       management is off for this subtest and an Interrupt Stack not valid
0419      ;       HALT should occur (code = 4).
0419      ;
0419      ; Subtest steps:
0419      ;
0419      ;       SUBTEST IS points to NXM
0419      ;       :  Initialize Flags
0419      ;       :  Clear RPB from location 0 in memory
0419      ;       :  Setup code to set done flag at loc 100
0419      ;       :  Save current IS SP
0419      ;       :  Setup a jump to RENTRY point at next loc
0419      ;       :  Get XFC vector to handle on interrupt stack
0419      ;       :  Execute XFC instruction
0419      ;       :  IF not on the interrupt stack
0419      ;       :  :  THEN
0419      ;       :  :  Report system error
0419      ;       :  :  Abort program
0419      ;       :  ENDIF
0419      ;       :  Release XFC vector
0419      ;       :  Get XFC vector for service on the interrupt stack
0419      ;       :  IF operator present
0419      ;       :  :  THEN
0419      ;       :  :  Print instructions to operator
0419      ;       :  ENDIF
0419      ;       :  IF operator NOT present
0419      ;       :  :  THEN
0419      ;       :  :  Setup RPB at loc 0 pointing to loc 100
0419      ;       :  ENDIF
0419      ;       :  Point interrupt stack pointer to NXM
0419      ;       :  Setup all expected data
0419      ;       :  Load GPR's with data
0419      ;       :  Set ready to execute flag
0419      ;       :  Execute XFC to cause exception which will write PC and PSL on IS
0419      ;       :  (this should cause an interrupt stack not valid HALT)
0419      ;       :  IF XFC not executed (will continue in line execution and get here)
0419      ;       :  :  THEN
0419      ;       :  :  Set XFC execution error flag
0419      ;       :  ENDIF
0419      ;       :  RENTRY point (restart will enter here)
0419      ;       :  Read PSL
0419      ;       :  IF on IS stack as expected
0419      ;       :  :  THEN
0419      ;       :  :  Restore IS SP
0419      ;       :  ENDIF
0419      ;       :  Go check FLAGS, PSL, KSP, ISP, SP and GPR's for errors
0419      ;       :  IF any errors
0419      ;       :  :  THEN
0419      ;       :  :  Report errors
0419      ;       :  ENDIF
```

```
                               0419   925 ;          :  IF fatal error
                               0419   926 ;          :  :  THEN
                               0419   927 ;          :  :  Abort test
                               0419   928 ;          :  ENDIF
                               0419   929 ;          :  Release XFC vector
                               0419   930 ;          ENDSUBTEST IS points to NXM
                               0419   931 ;
                               0419   932 ;-
                               0419   933
                               0419   934            $DS_BGNSUB
                               0419       T4_S3::
   00010030 9F   00000018'EF   FA  0419                      CALLG    $S$, a#DS$BGNSUB
                 00000BF8'EF   16  0424   935        JSB     CLEAR_RPB                ; disable restart parameter block
                 00000C2A'EF   16  042A   936        JSB     CLEAR_BUFFER             ; clear data buffers
                 0000003C'EF   D4  0430   937        CLRL    L_FLAGS                  ; initialize flags
                                   0436   938        $DS_MMON_S
         00010150 9F   00      FB  0436                      CALLS    #0, a#DS$MMON
                                   043D   939        $DS_SETVEC_S     VECTOR=#SCB$L_OPCCUS,SRVADR=RENTRY4,CODE=#1
                              01   DD  043D                  PUSHL    #1
                 00000450'EF   DF  043F                      PUSHAL   RENTRY4
                          14   DD  0445                      PUSHL    #SCB$L_OPCCUS
         00010160 9F   03      FB  0447                      CALLS    #3, a#DS$SETVEC
                               FC  044E   940 1$:    XFC                              ; get onto the IS
                                   044F   941        .ALIGN  LONG
                                   0450   942 RENTRY4:
                 00000000'EF   DC  0450   943        MOVPSL  L_ACT_PSL
       1A 00000000'EF   1A     E0  0456   944        BBS     #PSL$V_IS,L_ACT_PSL,2$
                                   045E   945        $DS_ERRHARD_S    UNIT=#0,MSGADR=T_NOT_ON_IS
                          00   DD  045E                      PUSHL    #0
                 0000089B'EF   DF  0460                      PUSHAL   T_NOT_ON_IS
                          00   DD  0466                      PUSHL    #0
                          01   DD  0468                      PUSHL    #$ER
                                   046A       ;;; TEST 4, SUBTEST 3, ERROR 1
         000100D0 9F   04      FB  046A                      CALLS    #$$M, a#DS$ERRHARD
                                   0471   946        $DS_ABORT
         00010020 9F   6C      FA  0471                      CALLG    (AP), a#DS$ABORT
                                   0478   947 2$:    $DS_CLRVEC_S     VECTOR=#SCB$L_OPCCUS
                          14   DD  0478                      PUSHL    #SCB$L_OPCCUS
         00010168 9F   01      FB  047A                      CALLS    #1, a#DS$CLRVEC
                                   0481   948
                                   0481   949        $DS_SETVEC_S     VECTOR=#SCB$L_OPCCUS,SRVADR=XFC_SERVICE,CODE=#1
                          01   DD  0481                      PUSHL    #1
                 00000CD8'EF   DF  0483                      PUSHAL   XFC_SERVICE
                          14   DD  0489                      PUSHL    #SCB$L_OPCCUS
         00010160 9F   03      FB  048B                      CALLS    #3, a#DS$SETVEC
   00000100'EF   00000040'EF   7D  0492   950 3$:    MOVQ    JUMP,RESTART             ; setup code to renter after halt
   00000108'EF   00000048'EF   7D  049D   951        MOVQ    JUMP+8,RESTART+8         ; load rest of rentry code
   00000004'EF   00000530'EF   DE  04A8   952        MOVAL   RENTRY5,RENTRY_POINT     ; load rentry address
         00000020'EF   5E      D0  04B3   953        MOVL    SP,L_CUR_ISP             ; read the current IS SP
       08 0000FE00'EF   0C     E0  04BA   954        BBS     #DSA$V_OPER,DSA$GL_FLAGS,4$ ; br = operator present, manual test
                 00000C00'EF   16  04C2   955        JSB     SETUP_RPB                ; no operator, use auto restart
                          22   11  04C8   956        BRB     5$                       ; skip over message
                                   04CA   957 4$:    $DS_PRINTF_S     FORMAT=T_INSTRUCTIONS,P0=#6$,P1=#CODE_04
                          04   DD  04CA                      PUSHL    #CODE_04
                 00000528'8F   DD  04CC                      PUSHL    #6$
                 000001DC'EF    9F  04D2                          PUSHAB  T_INSTRUCTIONS
           000100F0 9F    03    FB  04D8                          CALLS   #$$N, a#DS$PRINTF
                                    04DF    958           $DS_PRINTF_S    FORMAT=T_CONTINUE
                 00000212'EF    9F  04DF                          PUSHAB  T_CONTINUE
           000100F0 9F    01    FB  04E5                          CALLS   #$$N, a#DS$PRINTF
   00000000'EF   FFFFFFF8 8F    D0  04EC    959 5$:       MOVL    #-8,L_EXP_SP                 ; load expected SP
   00000000'EF   00000000'EF    D0  04F7    960           MOVL    L_KSP,L_EXP_KSP              ; load expected KSP
                 00000000'EF    D4  0502    961           CLRL    L_EXP_ISP                    ; ISP unpredictable if on int. stack
   00000000'EF   00000987'EF    D0  0508    962           MOVL    DATA_3,L_EXP_PSL             ; load expected PSL
                 00000C42'EF    16  0513    963           JSB     LOAD_GPR                     ; load GPR'S and read AP & FP
           5E    FFFFFFFC 8F    D0  0519    964           MOVL    #-4,SP                       ; point IS sp to a NXM address
           0000003C'EF    01    C8  0520    965           BISL2   #M_READY,L_FLAGS             ; set ready to execute flag
                                01  0527    966           NOP                                  ; wait a little
                                FC  0528    967 6$:       XFC                                  ; execute XFC instruction to cause
                                    0529    968                                                ; exception which will push on IS
                                    0529    969                                                ; and cause a not valid halt
           0000003C'EF    04    C8  0529    970           BISL2   #M_EXECUTE_ERR,L_FLAGS       ; if no exception occurs will get here
                                    0530    971
                                    0530    972 RENTRY5:                                       ; will renter here when completed
                                    0530    973
        11 00000000'EF    1A    E1  0530    974           BBC     #PSL$V_IS,L_ACT_PSL,8$       ; br = not on IS as expected
           5E    00000020'EF    D0  0538    975           MOVL    L_CUR_ISP,SP                 ; reset IS SP
           6E    00000547'8F    D0  053F    976           MOVL    #7$,(SP)                     ; setup return address
                                02  0546    977           REI                                  ; get off interrupt stack
   00000000'8F   00000020'EF    DA  0549    979 8$:       MTPR    L_CUR_ISP,#PR$_ISP           ; load IS SP into IPR
                 00000CE5'EF    16  0554    981 9$:       JSB     ERROR_CHECK                  ; check for errors (GPR's,AP,FP,PSL)
        28 0000003C'EF    0C    E1  055A    982           BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                 00000350'8F    DD  0562                          PUSHL   #T_NXMIS_ERR
                 0000099B'EF    DF  0568                          PUSHAL  ERR_MESSAGE
                          00    DD  056E                          PUSHL   #0
                          00    DD  0570                          PUSHL   #0
                          02    DD  0572                          PUSHL   #$ER
                                    0574          ;;; TEST 4, SUBTEST 3, ERROR 2
           000100D0 9F    05    FB  0574                          CALLS   #$$M, a#DS$ERRHARD
        07 0000003C'EF    08    E1  057B    984           BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                                    0583    985           $DS_ABORT
           00010020 9F    6C    FA  0583                          CALLG   (AP), a#DS$ABORT
                                    058A    986 10$:      $DS_MMOFF_S
           00010158 9F    00    FB  058A                          CALLS   #0, a#DS$MMOFF
                                    0591    987           $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                          14    DD  0591                          PUSHL   #SCB$L_OPCCUS
           00010168 9F    01    FB  0593                          CALLS   #1, a#DS$CLRVEC
                                    059A    988           $DS_ENDSUB
                                    059A          T4_S3_X:
   00010038 9F   00000018'EF    FA  059A                          CALLG   $S$, a#DS$ENDSUB
                                    05A5    989           $DS_PAGE
```

```
05A5                 .SBTTL  SUBTEST 4.4:  CHMU instruction with PSL<IS>=1
05A5      ;+
05A5      ; Subtest description:
05A5      ;
05A5      ;      This subtest checks that a CHMU instruction with PSL<IS>=1, will cause
05A5      ;      the processor to HALT (code = A).
05A5      ;
05A5      ; Subtest steps:
05A5      ;
05A5      ;      SUBTEST CHMU instruction with PSL<IS>=1
05A5      ;      :  Initialize Flags
05A5      ;      :  Clear RPB from location 0 in memory
05A5      ;      :  Get XFC vector to handle exception on interrupt stack
05A5      ;      :  Get CHMU vector in case CHMU instruction does execute
05A5      ;      :  Setup code to set done flag at loc 100
05A5      ;      :  Setup code to jump to READ_GPR point in next loc
05A5      ;      :  IF operator present
05A5      ;      :  :  THEN
05A5      ;      :  :  Print restart instructions to operator
05A5      ;      :  ENDIF
05A5      ;      :  IF operator NOT present
05A5      ;      :  :  THEN
05A5      ;      :  :  Setup RPB at loc 0 pointing to loc 100
05A5      ;      :  ENDIF
05A5      ;      :  Execute an XFC instruction to get on interrupt stack
05A5      ;      :  Read current PSL
05A5      ;      :  IF NOT on interrupt stack
05A5      ;      :  :  THEN
05A5      ;      :  :  Report wrong stack error
05A5      ;      :  :  Abort test
05A5      ;      :  ENDIF
05A5      ;      :  Setup all expected data
05A5      ;      :  Load GPR's with data
05A5      ;      :  Set ready to execute flag
05A5      ;      :  Execute CHMU to cause processor to HALT
05A5      ;      :  (this should cause a CHMU on interrupt stack HALT)
05A5      ;      :  IF CHMU does not caues an exception
05A5      ;      :  :  THEN
05A5      ;      :  :  Set execution error flag
05A5      ;      :  ENDIF
05A5      ;      :  RENTRY point (restart enters here from exception)
05A5      ;      :  Go check FLAGS, PSL, KSP, ISP, SP and GPR's for errors
05A5      ;      :  IF any errors
05A5      ;      :  :  THEN
05A5      ;      :  :  Report errors
05A5      ;      :  ENDIF
05A5      ;      :  IF fatal error
05A5      ;      :  :  THEN
05A5      ;      :  :  Abort test
05A5      ;      :  ENDIF
05A5      ;      :  Modify PC on stack to point to second RENTRY point
05A5      ;      :  Execute REI to get back to correct stack
05A5      ;      :  Second RENTRY point
05A5      ;      :  Read PSL
05A5      ;      :  IF not on kernel stack
```

```
                                     05A5  1046 ;       :  :  THEN
                                     05A5  1047 ;       :  :  Report incorrect stack error
                                     05A5  1048 ;       :  :  Abort test
                                     05A5  1049 ;       :  ENDIF
                                     05A5  1050 ;       :  Release interval timer vector
                                     05A5  1051 ;       :  Release CHMU vector
                                     05A5  1052 ;       :  Release XFC vector
                                     05A5  1053 ;       ENDSUBTEST CHMU instruction with PSL<IS>=1
                                     05A5  1054 ;
                                     05A5  1055 ;-
                                     05A5  1056
                                     05A5  1057         $DS_BGNSUB
                                     05A5       T4_S4::
     00010030 9F   00000024'EF   FA  05A5                       CALLG    $S$, @#DS$BGNSUB
                   00000BF8'EF   16  05B0  1058         JSB     CLEAR_RPB                ; disable restart parameter block
                   00000C2A'EF   16  05B6  1059         JSB     CLEAR_BUFFER             ; clear data buffers
                   0000003C'EF   D4  05BC  1060         CLRL    L_FLAGS                  ; initialize flags
                                     05C2  1061         $DS_SETVEC_S    VECTOR=#SCB$L_OPCCUS,SRVADR=RENTRY6,CODE=#1
                            01   DD  05C2                       PUSHL   #1
                   000005EC'EF   DF  05C4                       PUSHAL  RENTRY6
                            14   DD  05CA                       PUSHL   #SCB$L_OPCCUS
              00010160 9F   03   FB  05CC                       CALLS   #3, @#DS$SETVEC
                                     05D3  1062 1$:     $DS_SETVEC_S    VECTOR=#SCB$L_CHMU,SRVADR=CHMX_SERVICE
                            00   DD  05D3                       PUSHL   #0
                   00000CC4'EF   DF  05D5                       PUSHAL  CHMX_SERVICE
                   0000004C 8F   DD  05DB                       PUSHL   #SCB$L_CHMU
              00010160 9F   03   FB  05E1                       CALLS   #3, @#DS$SETVEC
                                     05E8  1063
                                 FC  05E8  1064 2$:     XFC                              ; execute XFC instruction to cause
                                     05E9  1065                                          ; exception which will get on IS
                                     05E9  1066         .ALIGN  LONG
                                     05EC  1067 RENTRY6:
                   00000000'EF   DC  05EC  1068         MOVPSL  L_ACT_PSL                ; read PSL
           1A 00000000'EF   1A   E0  05F2  1069         BBS     #PSL$V_IS,L_ACT_PSL,3$   ; br = on interrupt stack
                                     05FA  1070         $DS_ERRHARD_S   UNIT=#0,MSGADR=T_NOT_ON_IS
                            00   DD  05FA                       PUSHL   #0
                   0000089B'EF   DF  05FC                       PUSHAL  T_NOT_ON_IS
                            00   DD  0602                       PUSHL   #0
                            01   DD  0604                       PUSHL   #$ER
                                     0606       ;;; TEST 4, SUBTEST 4, ERROR 1
              000100D0 9F   04   FB  0606                       CALLS   #$$M, @#DS$ERRHARD
                                     060D  1071         $DS_ABORT
              00010020 9F   6C   FA  060D                       CALLG   (AP), @#DS$ABORT
     00000100'EF   00000040'EF   7D  0614  1072 3$:     MOVQ    JUMP,RESTART             ; setup code to renter after halt
     00000108'EF   00000048'EF   7D  061F  1073         MOVQ    JUMP+8,RESTART+8         ; load rest of rentry code
     00000004'EF   000006AD'EF   DE  062A  1074         MOVAL   RENTRY7,RENTRY_POINT     ; load rentry address
           08 0000FE00'EF   0C   E0  0635  1075         BBS     #DSA$V_OPER,DSA$GL_FLAGS,4$ ; br = operator present
                   00000C00'EF   16  063D  1076         JSB     SETUP_RPB                ; no operator, use auto restart
                            22   11  0643  1077         BRB     5$                       ; skip over message
                                     0645  1078 4$:     $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#6$,P1-#CODE_0A
                            0A   DD  0645                       PUSHL   #CODE_0A
                   000006A4'8F   DD  0647                       PUSHL   #6$
                   000001DC'EF   9F  064D                       PUSHAB  T_INSTRUCTIONS
              000100F0 9F   03   FB  0653                       CALLS   #$$N, @#DS$PRINTF
                                     065A  1079         $DS_PRINTF_S    FORMAT=T_CONTINUE
              00000212'EF    9F  065A                      PUSHAB  T_CONTINUE
          000100F0 9F    01  FB  0660                      CALLS   #$$N, a#DS$PRINTF
          00000020'EF    5E  D0  0667  1080 5$:     MOVL   SP,L_CUR_ISP
00000000'EF   00000000'EF    08  C3  066E  1081            SUBL3  #8,L_ISP,L_EXP_SP          ; load expected SP
    00000000'EF   00000000'EF    D0  067A  1082            MOVL   L_KSP,L_EXP_KSP            ; load expected KSP
          00000000'EF    00  D0  0685  1083            MOVL   #0,L_EXP_ISP               ; ISP unpredictable if on int. stack
    00000000'EF   0000098B'EF    D0  068C  1084            MOVL   DATA_4,L_EXP_PSL           ; load expected PSL
              00000C42'EF    16  0697  1085            JSB    LOAD_GPR                   ; load GPR's with data
          0000003C'EF    01  C8  069D  1086            BISL2  #M_READY,L_FLAGS           ; set ready to execute flag
                         00  BF  06A4  1087 6$:     CHMU   #0                         ; execute change mode instruction
          0000003C'EF    04  C8  06A6  1088            BISL2  #M_EXECUTE_ERR,L_FLAGS     ; if no exception occurs will get here
                             06AD  1089 RENTRY7:                                      ; will renter here when completed
       11 00000000'EF    1A  E1  06AD  1090            BBC    #PSL$V_IS,L_ACT_PSL,8$     ; br = not on IS as expected
          5E    00000020'EF  D0  06B5  1091            MOVL   L_CUR_ISP,SP               ; reset IS SP
          6E    000006C4'8F  D0  06BC  1092            MOVL   #7$,(SP)                   ; setup return address
                         02  06C3  1093            REI                               ; get off interrupt stack
                         0B  11  06C4  1094 7$:     BRB    9$                         ; skip IPR load
    00000000'8F   00000020'EF    DA  06C6  1095 8$:     MTPR   L_CUR_ISP,#PR$_ISP         ; load IS SP into IPR
              00000CE5'EF    16  06D1  1096 9$:     JSB    ERROR_CHECK                ; check for errors (GPR's,AP,FP,PSL)
       28 0000003C'EF    0C  E1  06D7  1097            BBC    #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                             06DF  1098            $DS_ERRHARD_S   UNIT=#0,PRLINK=ERR_MESSAGE,P1=#T_CHMU_ERR
              0000038D'8F    DD  06DF                      PUSHL   #T_CHMU_ERR
              0000099B'EF    DF  06E5                      PUSHAL  ERR_MESSAGE
                         00  DD  06EB                      PUSHL   #0
                         00  DD  06ED                      PUSHL   #0
                         02  DD  06EF                      PUSHL   #$ER
                             06F1       ;;; TEST 4, SUBTEST 4, ERROR 2
          000100D0 9F    05  FB  06F1                      CALLS   #$$M, a#DS$ERRHARD
       07 0000003C'EF    08  E1  06F8  1099            BBC    #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                             0700  1100            $DS_ABORT
          00010020 9F    6C  FA  0700                      CALLG   (AP), a#DS$ABORT
                             0707  1101 10$:    $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                         14  DD  0707                      PUSHL   #SCB$L_OPCCUS
          00010168 9F    01  FB  0709                      CALLS   #1, a#DS$CLRVEC
                             0710  1102            $DS_CLRVEC_S    VECTOR=#SCB$L_CHMU
              0000004C 8F    DD  0710                      PUSHL   #SCB$L_CHMU
          00010168 9F    01  FB  0716                      CALLS   #1, a#DS$CLRVEC
                             071D  1103            $DS_ENDSUB
                             071D       T4_S4_X:
    00010038 9F   00000024'EF    FA  071D                      CALLG   $S$, a#DS$ENDSUB
                             0728  1104            $DS_PAGE
                             0728  1105
```

```
0728                  .SBTTL  SUBTEST 4.5:  CHMS instruction with PSL<IS>=1
0728  1108 ;+
0728  1109 ; Subtest description:
0728  1110 ;
0728  1111 ;          This subtest checks that a CHMS instruction with PSL<IS>=1, will cause
0728  1112 ;          the processor to HALT (code = A).
0728  1113 ;
0728  1114 ; Subtest steps:
0728  1115 ;
0728  1116 ;          SUBTEST CHMS instruction with PSL<IS>=1
0728  1117 ;          :  Initialize Flags
0728  1118 ;          :  Clear RPB from location 0 in memory
0728  1119 ;          :  Get XFC vector to handle exception on interrupt stack
0728  1120 ;          :  Get interval timer vector to prevent outside interrupts
0728  1121 ;          :  Get CHMS vector in case CHMS instruction does execute
0728  1122 ;          :  Setup code to set done flag at loc 100
0728  1123 ;          :  Setup code to jump to RENTRY point in next loc
0728  1124 ;          :  IF operator present
0728  1125 ;          :  :  THEN
0728  1126 ;          :  :  Print restart instructions to operator
0728  1127 ;          :  ENDIF
0728  1128 ;          :  IF operator NOT present
0728  1129 ;          :  :  THEN
0728  1130 ;          :  :  Setup RPB at loc 0 pointing to loc 100
0728  1131 ;          :  ENDIF
0728  1132 ;          :  Execute an XFC instruction to get on interrupt stack
0728  1133 ;          :  Read current PSL
0728  1134 ;          :  IF NOT on interrupt stack
0728  1135 ;          :  :  THEN
0728  1136 ;          :  :  Report wrong stack error
0728  1137 ;          :  :  Exit test
0728  1138 ;          :  ENDIF
0728  1139 ;          :  Set ready to execute flag
0728  1140 ;          :  Execute CHMS to cause processor to HALT
0728  1141 ;          :  (this should cause a CHMS on interrupt stack HALT)
0728  1142 ;          :  IF CHMS does not caues an exception
0728  1143 ;          :  :  THEN
0728  1144 ;          :  :  Set execption error flag
0728  1145 ;          :  ENDIF
0728  1146 ;          :  RENTRY point (restart enters here from exception)
0728  1147 ;          :  IF execption error flag set
0728  1148 ;          :  :  THEN
0728  1149 ;          :  :  Report CHMS execption error
0728  1150 ;          :  :  Exit test
0728  1151 ;          :  ENDIF
0728  1152 ;          :  IF CHMS instruction executed flag set
0728  1153 ;          :  :  THEN
0728  1154 ;          :  :  Report change mode error
0728  1155 ;          :  :  Exit test
0728  1156 ;          :  ENDIF
0728  1157 ;          :  IF done flag not set
0728  1158 ;          :  :  THEN
0728  1159 ;          :  :  Report done error
0728  1160 ;          :  :  Exit test
0728  1161 ;          :  ENDIF
```

```
                                         0728  1162 ;          :  Modify PC on stack to point to RENTRY2
                                         0728  1163 ;          :  Execute REI to get back to correct stack
                                         0728  1164 ;          :  RENTRY2 point
                                         0728  1165 ;          :  Read PSL
                                         0728  1166 ;          :  IF IS not clear
                                         0728  1167 ;          :  :  THEN
                                         0728  1168 ;          :  :  Report incorrect stack error in clean up
                                         0728  1169 ;          :  :  Exit test
                                         0728  1170 ;          :  ENDIF
                                         0728  1171 ;          :  Release interval timer vector
                                         0728  1172 ;          :  Release CHMS vector
                                         0728  1173 ;          :  Release XFC vector
                                         0728  1174 ;         ENDSUBTEST CHMS instruction with PSL<IS>=1
                                         0728  1175 ;
                                         0728  1176 ;-
                                         0728  1177           $DS_BGNSUB
                                         0728      T4_S5::
         00010030 9F    00000030'EF   FA  0728                        CALLG   $S$, @#DS$BGNSUB
                        00000BF8'EF   16  0733  1178          JSB     CLEAR_RPB                 ; disable restart parameter block
                        00000C2A'EF   16  0739  1179          JSB     CLEAR_BUFFER              ; clear data buffers
                        0000003C'EF   D4  073F  1180          CLRL    L_FLAGS                   ; initialize flags
                                          0745  1181          $DS_SETVEC_S    VECTOR=#SCB$L_OPCCUS,SRVADR=RENTRY8,CODE=#1
                                 01   DD  0745                        PUSHL   #1
                        0000076C'EF   DF  0747                        PUSHAL  RENTRY8
                                 14   DD  074D                        PUSHL   #SCB$L_OPCCUS
             00010160 9F         03   FB  074F                        CALLS   #3, @#DS$SETVEC
                                          0756  1182 1$:      $DS_SETVEC_S    VECTOR=#SCB$L_CHMS,SRVADR=CHMX_SERVICE
                                 00   DD  0756                        PUSHL   #0
                        00000CC4'EF   DF  0758                        PUSHAL  CHMX_SERVICE
                        00000048 8F   DD  075E                        PUSHL   #SCB$L_CHMS
             00010160 9F         03   FB  0764                        CALLS   #3, @#DS$SETVEC
                                          076B  1183
                                      FC  076B  1184 2$:      XFC                               ; execute XFC instruction to cause
                                          076C  1185                                            ; exception which will get on IS
                                          076C  1186          .ALIGN  LONG
                                          076C  1187 RENTRY8:
                        00000000'EF   DC  076C  1188          MOVPSL  L_ACT_PSL                 ; read PSL
           1A 00000000'EF        1A   E0  0772  1189          BBS     #PSL$V_IS,L_ACT_PSL,3$    ; br = on interrupt stack
                                          077A  1190          $DS_ERRHARD_S   UNIT=#0,MSGADR=T_NOT_ON_IS
                                 00   DD  077A                        PUSHL   #0
                        0000089B'EF   DF  077C                        PUSHAL  T_NOT_ON_IS
                                 00   DD  0782                        PUSHL   #0
                                 01   DD  0784                        PUSHL   #$ER
                                          0786       ;;; TEST 4, SUBTEST 5, ERROR 1
             000100D0 9F         04   FB  0786                        CALLS   #$$M, @#DS$ERRHARD
                                          078D  1191          $DS_ABORT
             00010020 9F         6C   FA  078D                        CALLG   (AP), @#DS$ABORT
         00000100'EF    00000040'EF   7D  0794  1192 3$:      MOVQ    JUMP,RESTART              ; setup code to renter after halt
         00000108'EF    00000048'EF   7D  079F  1193          MOVQ    JUMP+8,RESTART+8          ; load rest of rentry code
         00000004'EF    0000082D'EF   DE  07AA  1194          MOVAL   RENTRY9,RENTRY_POINT      ; load rentry address
              08 0000FE00'EF     0C   E0  07B5  1195          BBS     #DSA$V_OPER,DSA$GL_FLAGS,4$ ; br = operator present
                        00000C00'EF   16  07BD  1196          JSB     SETUP_RPB                 ; no operator, use auto restart
                                 22   11  07C3  1197          BRB     5$                        ; skip over message
                                          07C5  1198 4$:      $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#6$,P1=#CODE_0A
                                 0A   DD  07C5                        PUSHL   #CODE_0A
```

```
                   00000824'8F   DD  07C7                       PUSHL   #6$
                   000001DC'EF   9F  07CD                       PUSHAB  T_INSTRUCTIONS
              000100F0 9F   03   FB  07D3                       CALLS   #$$N, @#DS$PRINTF
                                     07DA  1199         $DS_PRINTF_S    FORMAT=T_CONTINUE
                   00000212'EF   9F  07DA                       PUSHAB  T_CONTINUE
              000100F0 9F   01   FB  07E0                       CALLS   #$$N, @#DS$PRINTF
              00000020'EF   5E   D0  07E7  1200 5$:     MOVL    SP,L_CUR_ISP
00000000'EF   00000000'EF   08   C3  07EE  1201         SUBL3   #8,L_ISP,L_EXP_SP       ; load expected SP
     00000000'EF   00000000'EF   D0  07FA  1202         MOVL    L_KSP,L_EXP_KSP         ; load expected KSP
              00000000'EF   00   D0  0805  1203         MOVL    #0,L_EXP_ISP            ; ISP unpredictable if on int. stack
     00000000'EF   0000098B'EF   D0  080C  1204         MOVL    DATA_4,L_EXP_PSL        ; load expected PSL
                   00000C42'EF   16  0817  1205         JSB     LOAD_GPR                ; load GPR's with data
              0000003C'EF   01   C8  081D  1206         BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
                            00   BE  0824  1207 6$:     CHMS    #0                      ; execute change mode instruction
              0000003C'EF   04   C8  0826  1208         BISL2   #M_EXECUTE_ERR,L_FLAGS  ; if no exception occurs will get here
                                     082D  1209 RENTRY9:                                ; will renter here when completed
           11 00000000'EF   1A   E1  082D  1210         BBC     #PSL$V_IS,L_ACT_PSL,8$  ; br = not on IS as expected
              5E   00000020'EF   D0  0835  1211         MOVL    L_CUR_ISP,SP            ; reset IS SP
              6E   00000844'8F   D0  083C  1212         MOVL    #7$,(SP)                ; setup return address
                            02       0843  1213         REI                             ; get off interrupt stack
                            0B   11  0844  1214 7$:     BRB     9$                      ; skip IPR load
     00000000'8F   00000020'EF   DA  0846  1215 8$:     MTPR    L_CUR_ISP,#PR$_ISP      ; load IS SP into IPR
                   00000CE5'EF   16  0851  1216 9$:     JSB     ERROR_CHECK             ; check for errors (GPR's,AP,FP,PSL)
           28 0000003C'EF   0C   E1  0857  1217         BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                                     085F  1218         $DS_ERRHARD_S   UNIT=#0,PRLINK=ERR_MESSAGE,P1=#T_CHMS_ERR
                   000003B5'8F   DD  085F                       PUSHL   #T_CHMS_ERR
                   0000099B'EF   DF  0865                       PUSHAL  ERR_MESSAGE
                            00   DD  086B                       PUSHL   #0
                            00   DD  086D                       PUSHL   #0
                            02   DD  086F                       PUSHL   #$ER
                                     0871          ;;; TEST 4, SUBTEST 5, ERROR 2
              000100D0 9F   05   FB  0871                       CALLS   #$$M, @#DS$ERRHARD
           07 0000003C'EF   08   E1  0878  1219         BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                                     0880  1220         $DS_ABORT
              00010020 9F   6C   FA  0880                       CALLG   (AP), @#DS$ABORT
                                     0887  1221 10$:    $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                            14   DD  0887                       PUSHL   #SCB$L_OPCCUS
              00010168 9F   01   FB  0889                       CALLS   #1, @#DS$CLRVEC
                                     0890  1222         $DS_CLRVEC_S    VECTOR=#SCB$L_CHMS
                   00000048 8F   DD  0890                       PUSHL   #SCB$L_CHMS
              00010168 9F   01   FB  0896                       CALLS   #1, @#DS$CLRVEC
                                     089D  1223         $DS_ENDSUB
                                     089D          T4_S5_X:
     00010038 9F   00000030'EF   FA  089D                       CALLG   $S$, @#DS$ENDSUB
                                     08A8  1224         $DS_PAGE
```

```
08A8                 .SBTTL  SUBTEST 4.6:  CHME instruction with PSL<IS>=1
08A8       ;+
08A8       ; Subtest description:
08A8       ;
08A8       ;       This subtest checks that a CHME instruction with PSL<IS>=1, will cause
08A8       ;       the processor to HALT (code = A).
08A8       ;
08A8       ; Subtest steps:
08A8       ;       SUBTEST CHME instruction with PSL<IS>=1
08A8       ;       :  Initialize Flags
08A8       ;       :  Clear RPB from location 0 in memory
08A8       ;       :  Get XFC vector to handle exception on interrupt stack
08A8       ;       :  Get interval timer vector to prevent outside interrupts
08A8       ;       :  Get CHME vector in case CHME instruction does execute
08A8       ;       :  Setup code to set done flag at loc 100
08A8       ;       :  Setup code to jump to RENTRY point in next loc
08A8       ;       :  IF operator present
08A8       ;       :  :  THEN
08A8       ;       :  :  Print restart instructions to operator
08A8       ;       :  ENDIF
08A8       ;       :  IF operator NOT present
08A8       ;       :  :  THEN
08A8       ;       :  :  Setup RPB at loc 0 pointing to loc 100
08A8       ;       :  ENDIF
08A8       ;       :  Execute an XFC instruction to get on interrupt stack
08A8       ;       :  Read current PSL
08A8       ;       :  IF NOT on interrupt stack
08A8       ;       :  :  THEN
08A8       ;       :  :  Report wrong stack error
08A8       ;       :  :  Exit test
08A8       ;       :  ENDIF
08A8       ;       :  Set ready to execute flag
08A8       ;       :  Execute CHME to cause processor to HALT
08A8       ;       :  (this should cause a CHME on interrupt stack HALT)
08A8       ;       :  IF CHME does not caues an exception
08A8       ;       :  :  THEN
08A8       ;       :  :  Set execption error flag
08A8       ;       :  ENDIF
08A8       ;       :  RENTRY point (restart enters here from exception)
08A8       ;       :  IF execption error flag set
08A8       ;       :  :  THEN
08A8       ;       :  :  Report CHME execption error
08A8       ;       :  :  Exit test
08A8       ;       :  ENDIF
08A8       ;       :  IF CHME instruction executed flag set
08A8       ;       :  :  THEN
08A8       ;       :  :  Report change mode error
08A8       ;       :  :  Exit test
08A8       ;       :  ENDIF
08A8       ;       :  IF done flag not set
08A8       ;       :  :  THEN
08A8       ;       :  :  Report done error
08A8       ;       :  :  Exit test
08A8       ;       :  ENDIF
08A8       ;       :  Modify PC on stack to point to RENTRY2
```

```
                                08A8  1281 ;         :  Execute REI to get back to correct stack
                                08A8  1282 ;         :  RENTRY2 point
                                08A8  1283 ;         :  Read PSL
                                08A8  1284 ;         :  IF IS not clear
                                08A8  1285 ;         :  :  THEN
                                08A8  1286 ;         :  :  Report incorrect stack error in clean up
                                08A8  1287 ;         :  :  Exit test
                                08A8  1288 ;         :  ENDIF
                                08A8  1289 ;         :  Release interval timer vector
                                08A8  1290 ;         :  Release CHME vector
                                08A8  1291 ;         :  Release XFC vector
                                08A8  1292 ;         ENDSUBTEST CHME instruction with PSL<IS>=1
                                08A8  1293 ;
                                08A8  1294 ;-
                                08A8  1295           $DS_BGNSUB
                                08A8       T4_S6::
     00010030 9F    0000003C'EF   FA  08A8                    CALLG   $S$, a#DS$BGNSUB
                    00000BF8'EF   16  08B3  1296           JSB     CLEAR_RPB                ; disable restart parameter block
                    00000C2A'EF   16  08B9  1297           JSB     CLEAR_BUFFER             ; clear data buffers
                    0000003C'EF   D4  08BF  1298           CLRL    L_FLAGS                  ; initialize flags
                                      08C5  1299           $DS_SETVEC_S    VECTOR=#SCB$L_OPCCUS,SRVADR=RENTRY10,CODE=#1
                             01   DD  08C5                    PUSHL   #1
                    000008EC'EF   DF  08C7                    PUSHAL  RENTRY10
                             14   DD  08CD                    PUSHL   #SCB$L_OPCCUS
            00010160 9F      03   FB  08CF                    CALLS   #3, a#DS$SETVEC
                                      08D6  1300 1$:       $DS_SETVEC_S    VECTOR=#SCB$L_CHME,SRVADR=CHMX_SERVICE
                             00   DD  08D6                    PUSHL   #0
                    00000CC4'EF   DF  08D8                    PUSHAL  CHMX_SERVICE
                    00000044 8F   DD  08DE                    PUSHL   #SCB$L_CHME
            00010160 9F      03   FB  08E4                    CALLS   #3, a#DS$SETVEC
                                      08EB  1301
                                  FC  08EB  1302 2$:       XFC                              ; execute XFC instruction to cause
                                      08EC  1303                                            ; exception which will get on IS
                                      08EC  1304           .ALIGN  LONG
                                      08EC  1305 RENTRY10:
                    00000000'EF   DC  08EC  1306           MOVPSL  L_ACT_PSL                ; read PSL
          1A 00000000'EF     1A   E0  08F2  1307           BBS     #PSL$V_IS,L_ACT_PSL,3$   ; br = on interrupt stack
                                      08FA  1308           $DS_ERRHARD_S   UNIT=#0,MSGADR=T_NOT_ON_IS
                             00   DD  08FA                    PUSHL   #0
                    0000089B'EF   DF  08FC                    PUSHAL  T_NOT_ON_IS
                             00   DD  0902                    PUSHL   #0
                             01   DD  0904                    PUSHL   #$ER
                                      0906       ;;; TEST 4, SUBTEST 6, ERROR 1
            000100D0 9F      04   FB  0906                    CALLS   #$$M, a#DS$ERRHARD
                                      090D  1309           $DS_ABORT
            00010020 9F      6C   FA  090D                    CALLG   (AP), a#DS$ABORT
     00000100'EF    00000040'EF   7D  0914  1310 3$:       MOVQ    JUMP,RESTART             ; setup code to renter after halt
     00000108'EF    00000048'EF   7D  091F  1311           MOVQ    JUMP+8,RESTART+8         ; load rest of rentry code
     00000004'EF    000009AD'EF   DE  092A  1312           MOVAL   RENTRY11,RENTRY_POINT    ; load rentry address
          08 0000FE00'EF     0C   E0  0935  1313           BBS     #DSA$V_OPER,DSA$GL_FLAGS,4$ ; br = operator present
                    00000C00'EF   16  093D  1314           JSB     SETUP_RPB                ; no operator, use auto restart
                             22   11  0943  1315           BRB     5$                       ; skip over message
                                      0945  1316 4$:       $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#6$,P1=#CODE_0A
                             0A   DD  0945                    PUSHL   #CODE_0A
                    000009A4'8F   DD  0947                    PUSHL   #6$
```

```
              000001DC'EF    9F  094D                       PUSHAB  T_INSTRUCTIONS
          000100F0 9F    03  FB  0953                       CALLS   #$$N, @#DS$PRINTF
                                 095A  1317         $DS_PRINTF_S    FORMAT=T_CONTINUE
              00000212'EF    9F  095A                       PUSHAB  T_CONTINUE
          000100F0 9F    01  FB  0960                       CALLS   #$$N, @#DS$PRINTF
          00000020'EF    5E  D0  0967  1318 5$:     MOVL    SP,L_CUR_ISP
00000000'EF  00000000'EF  08  C3  096E  1319        SUBL3   #8,L_ISP,L_EXP_SP       ; load expected SP
     00000000'EF   00000000'EF  D0  097A  1320      MOVL    L_KSP,L_EXP_KSP         ; load expected KSP
              00000000'EF    00  D0  0985  1321     MOVL    #0,L_EXP_ISP            ; ISP unpredictable if on int. stack
     00000000'EF   0000098B'EF  D0  098C  1322      MOVL    DATA_4,L_EXP_PSL        ; load expected PSL
              00000C42'EF    16  0997  1323         JSB     LOAD_GPR                ; load GPR's with data
          0000003C'EF    01  C8  099D  1324         BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
                         00  BD  09A4  1325 6$:     CHME    #0                      ; execute change mode instruction
          0000003C'EF    04  C8  09A6  1326         BISL2   #M_EXECUTE_ERR,L_FLAGS  ; if no exception occurs will get here
                                 09AD  1327 RENTRY11:                               ; will renter here when completed
       11 00000000'EF    1A  E1  09AD  1328         BBC     #PSL$V_IS,L_ACT_PSL,8$  ; br = not on IS as expected
          5E   00000020'EF   D0  09B5  1329         MOVL    L_CUR_ISP,SP            ; reset IS SP
          6E   000009C4'8F   D0  09BC  1330         MOVL    #7$,(SP)                ; setup return address
                             02  09C3  1331         REI                             ; get off interrupt stack
                         0B  11  09C4  1332 7$:     BRB     9$                      ; skip IPR load
     00000000'8F   00000020'EF  DA  09C6  1333 8$:  MTPR    L_CUR_ISP,#PR$_ISP      ; load IS SP into IPR
              00000CE5'EF    16  09D1  1334 9$:     JSB     ERROR_CHECK             ; check for errors (GPR's,AP,FP,PSL)
       28 0000003C'EF    0C  E1  09D7  1335         BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                                 09DF  1336         $DS_ERRHARD_S   UNIT=#0,PRLINK=ERR_MESSAGE,P1=#T_CHME_ERR
              000003DD'8F    DD  09DF                       PUSHL   #T_CHME_ERR
              0000099B'EF    DF  09E5                       PUSHAL  ERR_MESSAGE
                         00  DD  09EB                       PUSHL   #0
                         00  DD  09ED                       PUSHL   #0
                         02  DD  09EF                       PUSHL   #$ER
                                 09F1           ;;; TEST 4, SUBTEST 6, ERROR 2
          000100D0 9F    05  FB  09F1                       CALLS   #$$M, @#DS$ERRHARD
       07 0000003C'EF    08  E1  09F8  1337         BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                                 0A00  1338         $DS_ABORT
          00010020 9F    6C  FA  0A00                       CALLG   (AP), @#DS$ABORT
                                 0A07  1339 10$:    $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                         14  DD  0A07                       PUSHL   #SCB$L_OPCCUS
          00010168 9F    01  FB  0A09                       CALLS   #1, @#DS$CLRVEC
                                 0A10  1340         $DS_CLRVEC_S    VECTOR=#SCB$L_CHME
              00000044 8F    DD  0A10                       PUSHL   #SCB$L_CHME
          00010168 9F    01  FB  0A16                       CALLS   #1, @#DS$CLRVEC
                                 0A1D  1341         $DS_ENDSUB
                                 0A1D           T4_S6_X:
     00010038 9F   0000003C'EF  FA  0A1D                    CALLG   $S$, @#DS$ENDSUB
                                 0A28  1342         $DS_PAGE
```

```
0A28            .SBTTL  SUBTEST 4.7:  CHMK instruction with PSL<IS>=1
0A28       ;+
0A28       ; Subtest description:
0A28       ;
0A28       ;        This subtest checks that a CHMK instruction with PSL<IS>=1, will cause
0A28       ;        the processor to HALT (code = A).
0A28       ;
0A28       ; Subtest steps:
0A28       ;
0A28       ;        SUBTEST CHMK instruction with PSL<IS>=1
0A28       ;        :  Initialize Flags
0A28       ;        :  Clear RPB from location 0 in memory
0A28       ;        :  Get XFC vector to handle exception on interrupt stack
0A28       ;        :  Get interval timer vector to prevent outside interrupts
0A28       ;        :  Get CHMK vector in case CHMK instruction does execute
0A28       ;        :  Setup code to set done flag at loc 100
0A28       ;        :  Setup code to jump to RENTRY point in next loc
0A28       ;        :  IF operator present
0A28       ;        :  :  THEN
0A28       ;        :  :  Print restart instructions to operator
0A28       ;        :  ENDIF
0A28       ;        :  IF operator NOT present
0A28       ;        :  :  THEN
0A28       ;        :  :  Setup RPB at loc 0 pointing to loc 100
0A28       ;        :  ENDIF
0A28       ;        :  Execute an XFC instruction to get on interrupt stack
0A28       ;        :  Read current PSL
0A28       ;        :  IF NOT on interrupt stack
0A28       ;        :  :  THEN
0A28       ;        :  :  Report wrong stack error
0A28       ;        :  :  Exit test
0A28       ;        :  ENDIF
0A28       ;        :  Set ready to execute flag
0A28       ;        :  Execute CHMK to cause processor to HALT
0A28       ;        :  (this should cause a CHMK on interrupt stack HALT)
0A28       ;        :  IF CHMK does not caues an exception
0A28       ;        :  :  THEN
0A28       ;        :  :  Set execption error flag
0A28       ;        :  ENDIF
0A28       ;        :  RENTRY point (restart enters here from exception)
0A28       ;        :  IF execption error flag set
0A28       ;        :  :  THEN
0A28       ;        :  :  Report CHMK execption error
0A28       ;        :  :  Exit test
0A28       ;        :  ENDIF
0A28       ;        :  IF CHMK instruction executed flag set
0A28       ;        :  :  THEN
0A28       ;        :  :  Report change mode error
0A28       ;        :  :  Exit test
0A28       ;        :  ENDIF
0A28       ;        :  IF done flag not set
0A28       ;        :  :  THEN
0A28       ;        :  :  Report done error
0A28       ;        :  :  Exit test
0A28       ;        :  ENDIF
```

```
                                     0A28  1399 ;       :  Modify PC on stack to point to RENTRY2
                                     0A28  1400 ;       :  Execute REI to get back to correct stack
                                     0A28  1401 ;       :  RENTRY2 point
                                     0A28  1402 ;       :  Read PSL
                                     0A28  1403 ;       :  IF IS not clear
                                     0A28  1404 ;       :  :  THEN
                                     0A28  1405 ;       :  :  Report incorrect stack error in clean up
                                     0A28  1406 ;       :  :  Exit test
                                     0A28  1407 ;       :  ENDIF
                                     0A28  1408 ;       :  Release interval timer vector
                                     0A28  1409 ;       :  Release CHMK vector
                                     0A28  1410 ;       :  Release XFC vector
                                     0A28  1411 ;       ENDSUBTEST CHMK instruction with PSL<IS>=1
                                     0A28  1412 ;
                                     0A28  1413 ;-
                                     0A28  1414         $DS_BGNSUB
                                     0A28       T4_S7::
     00010030 9F   00000048'EF   FA  0A28                       CALLG   $S$, @#DS$BGNSUB
                   00000BF8'EF   16  0A33  1415         JSB     CLEAR_RPB               ; disable restart parameter block
                   00000C2A'EF   16  0A39  1416         JSB     CLEAR_BUFFER            ; clear data buffers
                   0000003C'EF   D4  0A3F  1417         CLRL    L_FLAGS                 ; initialize flags
                                     0A45  1418         $DS_SETVEC_S    VECTOR=#SCB$L_OPCCUS,SRVADR=RENTRY12,CODE=#1
                            01   DD  0A45                       PUSHL   #1
                   00000A6C'EF   DF  0A47                       PUSHAL  RENTRY12
                            14   DD  0A4D                       PUSHL   #SCB$L_OPCCUS
              00010160 9F   03   FB  0A4F                       CALLS   #3, @#DS$SETVEC
                                     0A56  1419 1$:     $DS_SETVEC_S    VECTOR=#SCB$L_CHMK,SRVADR=CHMX_SERVICE
                            00   DD  0A56                       PUSHL   #0
                   00000CC4'EF   DF  0A58                       PUSHAL  CHMX_SERVICE
                   00000040 8F   DD  0A5E                       PUSHL   #SCB$L_CHMK
              00010160 9F   03   FB  0A64                       CALLS   #3, @#DS$SETVEC
                                     0A6B  1420
                                 FC  0A6B  1421 2$:     XFC                             ; execute XFC instruction to cause
                                     0A6C  1422                                         ; exception which will get on IS
                                     0A6C  1423         .ALIGN  LONG
                                     0A6C  1424 RENTRY12:
                   00000000'EF   DC  0A6C  1425         MOVPSL  L_ACT_PSL               ; read PSL
           1A 00000000'EF   1A   E0  0A72  1426         BBS     #PSL$V_IS,L_ACT_PSL,3$  ; br = on interrupt stack
                                     0A7A  1427         $DS_ERRHARD_S   UNIT=#0,MSGADR=T_NOT_ON_IS
                            00   DD  0A7A                       PUSHL   #0
                   0000089B'EF   DF  0A7C                       PUSHAL  T_NOT_ON_IS
                            00   DD  0A82                       PUSHL   #0
                            01   DD  0A84                       PUSHL   #$ER
                                     0A86       ;;; TEST 4, SUBTEST 7, ERROR 1
              000100D0 9F   04   FB  0A86                       CALLS   #$$M, @#DS$ERRHARD
                                     0A8D  1428         $DS_ABORT
              00010020 9F   6C   FA  0A8D                       CALLG   (AP), @#DS$ABORT
     00000100'EF   00000040'EF   7D  0A94  1429 3$:     MOVQ    JUMP,RESTART            ; setup code to renter after halt
     00000108'EF   00000048'EF   7D  0A9F  1430         MOVQ    JUMP+8,RESTART+8        ; load rest of rentry code
     00000004'EF   00000B2D'EF   DE  0AAA  1431         MOVAL   RENTRY13,RENTRY_POINT   ; load rentry address
           08 0000FE00'EF   0C   E0  0AB5  1432         BBS     #DSA$V_OPER,DSA$GL_FLAGS,4$ ; br = operator present
                   00000C00'EF   16  0ABD  1433         JSB     SETUP_RPB               ; no operator, use auto restart
                            22   11  0AC3  1434         BRB     5$                      ; skip over message
                                     0AC5  1435 4$:     $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#6$,P1=#CODE_0A
                            0A   DD  0AC5                       PUSHL   #CODE_0A
                   00000B24'8F    DD  0AC7                       PUSHL   #6$
                   000001DC'EF    9F  0ACD                       PUSHAB  T_INSTRUCTIONS
              000100F0 9F    03   FB  0AD3                       CALLS   #$$N, @#DS$PRINTF
                                      0ADA  1436         $DS_PRINTF_S    FORMAT=T_CONTINUE
                   00000212'EF    9F  0ADA                       PUSHAB  T_CONTINUE
              000100F0 9F    01   FB  0AE0                       CALLS   #$$N, @#DS$PRINTF
              00000020'EF    5E   D0  0AE7  1437 5$:     MOVL    SP,L_CUR_ISP
00000000'EF   00000000'EF    08   C3  0AEE  1438         SUBL3   #8,L_ISP,L_EXP_SP       ; load expected SP
     00000000'EF    00000000'EF    D0  0AFA  1439         MOVL    L_KSP,L_EXP_KSP         ; load expected KSP
              00000000'EF    00   D0  0B05  1440         MOVL    #0,L_EXP_ISP            ; ISP unpredictable if on int. stack
     00000000'EF    0000098B'EF    D0  0B0C  1441         MOVL    DATA_4,L_EXP_PSL        ; load expected PSL
                   00000C42'EF    16  0B17  1442         JSB     LOAD_GPR                ; load GPR's with data
              0000003C'EF    01   C8  0B1D  1443         BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
                            00   BC  0B24  1444 6$:     CHMK    #0                      ; execute change mode instruction
              0000003C'EF    04   C8  0B26  1445         BISL2   #M_EXECUTE_ERR,L_FLAGS  ; if no exception occurs will get here
                                      0B2D  1446 RENTRY13:                               ; will renter here when completed
           11 00000000'EF    1A   E1  0B2D  1447         BBC     #PSL$V_IS,L_ACT_PSL,8$  ; br = not on IS as expected
              5E    00000020'EF    D0  0B35  1448         MOVL    L_CUR_ISP,SP            ; reset IS SP
              6E    00000B44'8F    D0  0B3C  1449         MOVL    #7$,(SP)                ; setup return address
                                 02  0B43  1450         REI                             ; get off interrupt stack
                            0B   11  0B44  1451 7$:     BRB     9$                      ; skip IPR load
     00000000'8F    00000020'EF    DA  0B46  1452 8$:     MTPR    L_CUR_ISP,#PR$_ISP      ; load IS SP into IPR
                   00000CE5'EF    16  0B51  1453 9$:     JSB     ERROR_CHECK             ; check for errors (GPR's,AP,FP,PSL)
           28 0000003C'EF    0C   E1  0B57  1454         BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                                      0B5F  1455         $DS_ERRHARD_S   UNIT=#0,PRLINK=ERR_MESSAGE,P1=#T_CHMK_ERR
                   00000405'8F    DD  0B5F                       PUSHL   #T_CHMK_ERR
                   0000099B'EF    DF  0B65                       PUSHAL  ERR_MESSAGE
                            00   DD  0B6B                       PUSHL   #0
                            00   DD  0B6D                       PUSHL   #0
                            02   DD  0B6F                       PUSHL   #$ER
                                      0B71       ;;; TEST 4, SUBTEST 7, ERROR 2
              000100D0 9F    05   FB  0B71                       CALLS   #$$M, @#DS$ERRHARD
           07 0000003C'EF    08   E1  0B78  1456         BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                                      0B80  1457         $DS_ABORT
              00010020 9F    6C   FA  0B80                       CALLG   (AP), @#DS$ABORT
                                      0B87  1458 10$:    $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                            14   DD  0B87                       PUSHL   #SCB$L_OPCCUS
              00010168 9F    01   FB  0B89                       CALLS   #1, @#DS$CLRVEC
                                      0B90  1459         $DS_CLRVEC_S    VECTOR=#SCB$L_CHMK
                   00000040 8F    DD  0B90                       PUSHL   #SCB$L_CHMK
              00010168 9F    01   FB  0B96                       CALLS   #1, @#DS$CLRVEC
                                      0B9D  1460         $DS_ENDSUB
                                      0B9D       T4_S7_X:
     00010038 9F    00000048'EF    FA  0B9D                       CALLG   $S$, @#DS$ENDSUB
                                      0BA8  1461         $DS_CLRVEC_S    VECTOR=#SCB$L_TIMER
                   000000C0 8F    DD  0BA8                       PUSHL   #SCB$L_TIMER
              00010168 9F    01   FB  0BAE                       CALLS   #1, @#DS$CLRVEC
                                      0BB5  1462         $DS_ENDTEST
                       50    01   D0  0BB5                       MOVL    #1, R0                  ; NORMAL EXIT
                                      0BB8       TEST_004_X::
              00010058 9F    6E   FA  0BB8                       CALLG   (SP), @#DS$BREAK
                                 04  0BBF                       RET                             ; RETURN TO TEST SEQUENCER
                                      0BC0  1463         $DS_PAGE
                                      0BC0  1464
                                      0BC0  1465
```

```
                                     0BC0               .SBTTL  TEST 5:  PROCESSOR HALTS ARCH. UNDEFINED
                                 00000000                       .PSECT  TEST_005, PAGE, NOWRT
                                     0024
                                     0024  1468 ;++
                                     0024  1469 ; Test description:
                                     0024  1470 ;
                                     0024  1471 ;       This test checks the various ways the VAX-11/730 processor can
                                     0024  1472 ;       be halted in an architecturally undefined manner. This includes
                                     0024  1473 ;       change mode instructions with the vector specifying service on
                                     0024  1474 ;       the interrupt stack, invalid vector halt, SCBB read error halt,
                                     0024  1475 ;       double machine check halt, and transfer to NXM user WCS halt.
                                     0024  1476 ;
                                     0024  1477 ;       INSTRUCTIONS FOR EARLY ABORT:  If it is desired to exit the test
                                     0024  1478 ;       before completion, one must wait for the console prompt ( >>> )
                                     0024  1479 ;       after the halt occurs and type in the sequence D PC adr <CR>
                                     0024  1480 ;       C <CR>, where adr is the absolute address of the label
                                     0024  1481 ;       TEST_004_X:: in the program listing.
                                     0024  1482 ;
                                     0024  1483 ;--
                                     0024  1484
                                     0024  1485
                                     0024  1486         $DS_BGNTEST     <MANUAL,HALT,ALL>
                                     0024       DATA_005:
                           00000000  0024                       .LONG   0               ; TEST ARGUMENT TABLE TERMINATOR
                                     0028       TEST_005::
                               0000  0028                       .WORD   ^M<>            ; ENTRY MASK
                                     002A  1487         $DS_BOPER 5$                            ; Is operator present?
                            0C   E0  002A                       BBS     #DSA$V_OPER,-   ; BR IF OPERATOR
                14 0000FE00 9F       002C                               @#DSA$GL_FLAGS, 5$
                                     0032  1488         $DS_PRINTF_S    FORMAT=T_NO_OPER        ;5.2
                   0000004D'EF   9F  0032                       PUSHAB  T_NO_OPER
              000100F0 9F   01   FB  0038                       CALLS   #$$N, @#DS$PRINTF
                                     003F  1489         $DS_ABORT                               ; No. abort
              00010020 9F   6C   FA  003F                       CALLG   (AP), @#DS$ABORT
                                     0046  1490
                                     0046  1491 5$:     $DS_PRINTF_S    FORMAT=T_WARNING        ; print warning
                   00000262'EF   9F  0046                       PUSHAB  T_WARNING
              000100F0 9F   01   FB  004C                       CALLS   #$$N, @#DS$PRINTF
                                     0053  1492
                                     0053  1493
              00000000'EF   5E   D0  0053  1494         MOVL    SP,L_KSP                ; save initial KSP
     00000000'EF   00000000'8F   DB  005A  1495         MFPR    #PR$_ISP,L_ISP          ; save initial ISP
                   00000C2A'EF   16  0065  1496         JSB     CLEAR_BUFFER            ; clear buffer area
                                     006B  1497         $DS_SETVEC_S    VECTOR=#SCB$L_TIMER, SRVADR=TIMER_SERVICE
                            00   DD  006B                       PUSHL   #0
                   00000CB8'EF   DF  006D                       PUSHAL  TIMER_SERVICE
                   000000C0 8F   DD  0073                       PUSHL   #SCB$L_TIMER
              00010160 9F   03   FB  0079                       CALLS   #3, @#DS$SETVEC
                                     0080  1498 10$:
                                     0080  1499         $DS_PAGE
```

```
                                                 0080              .SBTTL  SUBTEST 5.1:  CHMU inst. with vector service on IS
                                                 0080   1502 ;+
                                                 0080   1503 ; Subtest description:
                                                 0080   1504 ;
                                                 0080   1505 ;         This subtest checks that a CHMU instruction with the SCB vector entry
                                                 0080   1506 ;         <1:0>=1 specifying service on the interrupt stack will cause
                                                 0080   1507 ;         the processor to HALT (code = B).
                                                 0080   1508 ;
                                                 0080   1509 ; Subtest steps:
                                                 0080   1510 ;
                                                 0080   1511 ;         SUBTEST CHMU instruction with vector service on IS
                                                 0080   1512 ;         :  Initialize Flags
                                                 0080   1513 ;         :  Get interval timer vector to prevent outside interrupts
                                                 0080   1514 ;         :  Get CHMU vector specifying service on IS
                                                 0080   1515 ;         :  Setup code to set done flag at loc 100
                                                 0080   1516 ;         :  Setup code to jump to RENTRY point in next loc
                                                 0080   1517 ;         :  Print restart instructions to operator
                                                 0080   1518 ;         :  Set ready to execute flag
                                                 0080   1519 ;         :  Execute CHMU to cause processor to HALT
                                                 0080   1520 ;         :  (this should cause a CHMU service on interrupt stack HALT)
                                                 0080   1521 ;         :  IF CHMU does not caues an exception
                                                 0080   1522 ;         :  :  THEN
                                                 0080   1523 ;         :  :  Set exception error flag
                                                 0080   1524 ;         :  ENDIF
                                                 0080   1525 ;         :  RENTRY point (restart enters here from exception)
                                                 0080   1526 ;         :  IF exception error flag set
                                                 0080   1527 ;         :  :  THEN
                                                 0080   1528 ;         :  :  Report CHMU exception error
                                                 0080   1529 ;         :  :  Exit test
                                                 0080   1530 ;         :  ENDIF
                                                 0080   1531 ;         :  IF CHMU instruction executed flag set
                                                 0080   1532 ;         :  :  THEN
                                                 0080   1533 ;         :  :  Report change mode error
                                                 0080   1534 ;         :  :  Exit test
                                                 0080   1535 ;         :  ENDIF
                                                 0080   1536 ;         :  IF done flag not set
                                                 0080   1537 ;         :  :  THEN
                                                 0080   1538 ;         :  :  Report done error
                                                 0080   1539 ;         :  :  Exit test
                                                 0080   1540 ;         :  ENDIF
                                                 0080   1541 ;         :  Read PSL
                                                 0080   1542 ;         :  IF IS not clear
                                                 0080   1543 ;         :  :  THEN
                                                 0080   1544 ;         :  :  Report incorrect stack error in clean up
                                                 0080   1545 ;         :  :  Exit test
                                                 0080   1546 ;         :  ENDIF
                                                 0080   1547 ;         :  Release interval timer vector
                                                 0080   1548 ;         :  Release CHMU vector
                                                 0080   1549 ;         ENDSUBTEST CHMU instruction with
                                                 0080   1550 ;
                                                 0080   1551 ;-
                                                 0080   1552              $DS_BGNSUB
                                                 0080          T5_S1::
       00010030 9F      00000054'EF      FA      0080                             CALLG   $S$, a#DS$BGNSUB
                        00000C2A'EF      16      008B   1553              JSB     CLEAR_BUFFER              ; clear data buffers
```

```
                   0000003C'EF   D4  0091  1554         CLRL    L_FLAGS                 ; initialize flags
                                     0097  1555 1$:     $DS_SETVEC_S    VECTOR=#SCB$L_CHMU,SRVADR=CHMX_SERVICE,CODE=#1
                            01   DD  0097                       PUSHL   #1
                   00000CC4'EF   DF  0099                       PUSHAL  CHMX_SERVICE
                   0000004C 8F   DD  009F                       PUSHL   #SCB$L_CHMU
              00010160 9F   03   FB  00A5                       CALLS   #3, @#DS$SETVEC
                                     00AC  1556
     00000100'EF   00000040'EF   7D  00AC  1557 3$:     MOVQ    JUMP,RESTART            ; setup code to renter after halt
     00000108'EF   00000048'EF   7D  00B7  1558         MOVQ    JUMP+8,RESTART+8        ; load rest of rentry code
     00000004'EF   0000013C'EF   DE  00C2  1559         MOVAL   RENTRY15,RENTRY_POINT   ; load rentry address
                                     00CD  1560 4$:     $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#6$,P1=#CODE_0B
                            0B   DD  00CD                       PUSHL   #CODE_0B
                   00000133'8F   DD  00CF                       PUSHL   #6$
                   000001DC'EF   9F  00D5                       PUSHAB  T_INSTRUCTIONS
              000100F0 9F   03   FB  00DB                       CALLS   #$$N, @#DS$PRINTF
                                     00E2  1561         $DS_PRINTF_S    FORMAT=T_CONTINUE
                   00000212'EF   9F  00E2                       PUSHAB  T_CONTINUE
              000100F0 9F   01   FB  00E8                       CALLS   #$$N, @#DS$PRINTF
     00000020'EF   00000000'EF   D0  00EF  1562 5$:     MOVL    L_ISP,L_CUR_ISP         ; save current IS SP
     00000000'EF   00000000'EF   D0  00FA  1563         MOVL    L_KSP,L_EXP_SP          ; load expected SP
     00000000'EF   00000000'EF   D0  0105  1564         MOVL    L_KSP,L_EXP_KSP         ; load expected KSP
     00000000'EF   00000000'EF   D0  0110  1565         MOVL    L_ISP,L_EXP_ISP         ; load expected ISP
     00000000'EF   0000097F'EF   D0  011B  1566         MOVL    DATA_1,L_EXP_PSL        ; load expected PSL
                   00000C42'EF   16  0126  1567         JSB     LOAD_GPR                ; load GPR's with data
              0000003C'EF   01   C8  012C  1568         BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
                            00   BF  0133  1569 6$:     CHMU    #0                      ; execute change mode instruction
              0000003C'EF   04   C8  0135  1570         BISL2   #M_EXECUTE_ERR,L_FLAGS  ; if no exception occurs will get here
                                     013C  1571 RENTRY15:                               ; will renter here when completed
           1C 00000000'EF   1A   E1  013C  1572         BBC     #PSL$V_IS,L_ACT_PSL,9$  ; br = not on IS as expected
              5E   00000020'EF   D0  0144  1573         MOVL    L_CUR_ISP,SP            ; reset IS SP
              6E   00000153'8F   D0  014B  1574         MOVL    #7$,(SP)                ; setup return address
                                 02  0152  1575         REI                             ; get off interrupt stack
                            0B   11  0153  1576 7$:     BRB     9$                      ; skip IPR load
     00000000'8F   00000020'EF   DA  0155  1577 8$:     MTPR    L_CUR_ISP,#PR$_ISP      ; load IS SP into IPR
                   00000CE5'EF   16  0160  1578 9$:     JSB     ERROR_CHECK             ; check for errors (GPR's,AP,FP,PSL)
           2C 0000003C'EF   0C   E1  0166  1579         BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                                     016E  1580         $DS_ERRHARD_S   UNIT=KA_UNIT_NO, PRLINK=ERR_MESSAGE,P1=#T_CHMUSCB_ERR
                   000004B5'8F   DD  016E                       PUSHL   #T_CHMUSCB_ERR
                   0000099B'EF   DF  0174                       PUSHAL  ERR_MESSAGE
                            00   DD  017A                       PUSHL   #0
                   00000000'EF   DD  017C                       PUSHL   KA_UNIT_NO
                            01   DD  0182                       PUSHL   #$ER
                                     0184       ;;; TEST 5, SUBTEST 1, ERROR 1
              000100D0 9F   05   FB  0184                       CALLS   #$$M, @#DS$ERRHARD
           07 0000003C'EF   08   E1  018B  1581         BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                                     0193  1582         $DS_ABORT
              00010020 9F   6C   FA  0193                       CALLG   (AP), @#DS$ABORT
                                     019A  1583 10$:    $DS_CLRVEC_S    VECTOR=#SCB$L_CHMU
                   0000004C 8F   DD  019A                       PUSHL   #SCB$L_CHMU
              00010168 9F   01   FB  01A0                       CALLS   #1, @#DS$CLRVEC
                                     01A7  1584         $DS_ENDSUB
                                     01A7       T5_S1_X:
     00010038 9F   00000054'EF   FA  01A7                       CALLG   $S$, @#DS$ENDSUB
                                     01B2  1585         $DS_PAGE
                                     01B2               .SBTTL  SUBTEST 5.2:  CHMS inst. with vector service on IS
                                     01B2       ;+
                                     01B2       ; Subtest description:
                                     01B2       ;
                                     01B2       ;       This subtest checks that a CHMS instruction with the SCB vector entry
                                     01B2       ;       <1:0>=1 specifying service on the interrupt stack will cause
                                     01B2       ;       the processor to HALT (code = B).
                                     01B2       ;
                                     01B2       ; Subtest steps:
                                     01B2       ;
                                     01B2       ;       SUBTEST CHMS instruction with vector service on IS
                                     01B2       ;       :  Initialize Flags
                                     01B2       ;       :  Get interval timer vector to prevent outside interrupts
                                     01B2       ;       :  Get CHMS vector specifying service on IS
                                     01B2       ;       :  Setup code to set done flag at loc 100
                                     01B2       ;       :  Setup code to jump to RENTRY point in next loc
                                     01B2       ;       :  Print restart instructions to operator
                                     01B2       ;       :  Set ready to execute flag
                                     01B2       ;       :  Execute CHMS to cause processor to HALT
                                     01B2       ;       :  (this should cause a CHMS service on interrupt stack HALT)
                                     01B2       ;       :  IF CHMS does not caues an exception
                                     01B2       ;       :  :  THEN
                                     01B2       ;       :  :  Set exception error flag
                                     01B2       ;       :  ENDIF
                                     01B2       ;       :  RENTRY point (restart enters here from exception)
                                     01B2       ;       :  IF exception error flag set
                                     01B2       ;       :  :  THEN
                                     01B2       ;       :  :  Report CHMS exception error
                                     01B2       ;       :  :  Exit test
                                     01B2       ;       :  ENDIF
                                     01B2       ;       :  IF CHMS instruction executed flag set
                                     01B2       ;       :  :  THEN
                                     01B2       ;       :  :  Report change mode error
                                     01B2       ;       :  :  Exit test
                                     01B2       ;       :  ENDIF
                                     01B2       ;       :  IF done flag not set
                                     01B2       ;       :  :  THEN
                                     01B2       ;       :  :  Report done error
                                     01B2       ;       :  :  Exit test
                                     01B2       ;       :  ENDIF
                                     01B2       ;       :  Read PSL
                                     01B2       ;       :  IF IS not clear
                                     01B2       ;       :  :  THEN
                                     01B2       ;       :  :  Report incorrect stack error in clean up
                                     01B2       ;       :  :  Exit test
                                     01B2       ;       :  ENDIF
                                     01B2       ;       :  Release interval timer vector
                                     01B2       ;       :  Release CHMS vector
                                     01B2       ;       ENDSUBTEST CHMS instruction with
                                     01B2       ;
                                     01B2       ;-
                                     01B2               $DS_BGNSUB
                                     01B2       T5_S2::
     00010030 9F   00000060'EF   FA  01B2                       CALLG   $S$, @#DS$BGNSUB
                   00000C2A'EF   16  01BD               JSB     CLEAR_BUFFER             ; clear data buffers
```

```
             0000003C'EF   D4  01C3  1640          CLRL    L_FLAGS                   ; initialize flags
                               01C9  1641 1$:      $DS_SETVEC_S    VECTOR=#SCB$L_CHMS,SRVADR=CHMX_SERVICE,CODE=#1
                     01    DD  01C9                        PUSHL   #1
             00000CC4'EF   DF  01CB                        PUSHAL  CHMX_SERVICE
             00000048 8F   DD  01D1                        PUSHL   #SCB$L_CHMS
      00010160 9F    03    FB  01D7                        CALLS   #3, @#DS$SETVEC
                               01DE  1642
00000100'EF  00000040'EF   7D  01DE  1643 3$:      MOVQ    JUMP,RESTART              ; setup code to renter after halt
00000108'EF  00000048'EF   7D  01E9  1644          MOVQ    JUMP+8,RESTART+8          ; load rest of rentry code
00000004'EF  0000026E'EF   DE  01F4  1645          MOVAL   RENTRY16,RENTRY_POINT     ; load rentry address
                               01FF  1646 4$:      $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#6$,P1=#CODE_0B
                     0B    DD  01FF                        PUSHL   #CODE_0B
             00000265'8F   DD  0201                        PUSHL   #6$
             000001DC'EF   9F  0207                        PUSHAB  T_INSTRUCTIONS
      000100F0 9F    03    FB  020D                        CALLS   #$$N, @#DS$PRINTF
                               0214  1647          $DS_PRINTF_S    FORMAT=T_CONTINUE
             00000212'EF   9F  0214                        PUSHAB  T_CONTINUE
      000100F0 9F    01    FB  021A                        CALLS   #$$N, @#DS$PRINTF
00000020'EF  00000000'EF   D0  0221  1648 5$:      MOVL    L_ISP,L_CUR_ISP           ; save current IS SP
00000000'EF  00000000'EF   D0  022C  1649          MOVL    L_KSP,L_EXP_SP            ; load expected SP
00000000'EF  00000000'EF   D0  0237  1650          MOVL    L_KSP,L_EXP_KSP           ; load expected KSP
00000000'EF  00000000'EF   D0  0242  1651          MOVL    L_ISP,L_EXP_ISP           ; load expected ISP
00000000'EF  0000097F'EF   D0  024D  1652          MOVL    DATA_1,L_EXP_PSL          ; load expected PSL
             00000C42'EF   16  0258  1653          JSB     LOAD_GPR                  ; load GPR's with data
      0000003C'EF    01    C8  025E  1654          BISL2   #M_READY,L_FLAGS          ; set ready to execute flag
                     00    BE  0265  1655 6$:      CHMS    #0                        ; execute change mode instruction
      0000003C'EF    04    C8  0267  1656          BISL2   #M_EXECUTE_ERR,L_FLAGS    ; if no exception occurs will get here
                               026E  1657 RENTRY16:                                  ; will renter here when completed
   1C 00000000'EF    1A    E1  026E  1658          BBC     #PSL$V_IS,L_ACT_PSL,9$    ; br = not on IS as expected
      5E     00000020'EF   D0  0276  1659          MOVL    L_CUR_ISP,SP              ; reset IS SP
      6E     00000285'8F   D0  027D  1660          MOVL    #7$,(SP)                  ; setup return address
                     02    02  0284  1661          REI                               ; get off interrupt stack
                     0B    11  0285  1662 7$:      BRB     9$                        ; skip IPR load
00000000'8F  00000020'EF   DA  0287  1663 8$:      MTPR    L_CUR_ISP,#PR$_ISP        ; load IS SP into IPR
             00000CE5'EF   16  0292  1664 9$:      JSB     ERROR_CHECK               ; check for errors (GPR's,AP,FP,PSL)
   2C 0000003C'EF    0C    E1  0298  1665          BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                               02A0  1666          $DS_ERRHARD_S   UNIT=KA_UNIT_NO, PRLINK=ERR_MESSAGE,P1=#T_CHMSSCB_ERR
             000004F0'8F   DD  02A0                        PUSHL   #T_CHMSSCB_ERR
             0000099B'EF   DF  02A6                        PUSHAL  ERR_MESSAGE
                     00    DD  02AC                        PUSHL   #0
             00000000'EF   DD  02AE                        PUSHL   KA_UNIT_NO
                     01    DD  02B4                        PUSHL   #$ER
                               02B6            ;;; TEST 5, SUBTEST 2, ERROR 1
      000100D0 9F    05    FB  02B6                        CALLS   #$$M, @#DS$ERRHARD
   07 0000003C'EF    08    E1  02BD  1667          BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                               02C5  1668          $DS_ABORT
      00010020 9F    6C    FA  02C5                        CALLG   (AP), @#DS$ABORT
                               02CC  1669 10$:     $DS_CLRVEC_S    VECTOR=#SCB$L_CHMS
             00000048 8F   DD  02CC                        PUSHL   #SCB$L_CHMS
      00010168 9F    01    FB  02D2                        CALLS   #1, @#DS$CLRVEC
                               02D9  1670          $DS_ENDSUB
                               02D9            T5_S2_X:
00010038 9F  00000060'EF   FA  02D9                        CALLG   $S$, @#DS$ENDSUB
                               02E4  1671          $DS_PAGE
```

```
                                     02E4                   .SBTTL  SUBTEST 5.3:  CHME inst. with vector service on IS
                                     02E4  1674 ;+
                                     02E4  1675 ; Subtest description:
                                     02E4  1676 ;
                                     02E4  1677 ;       This subtest checks that a CHME instruction with the SCB vector entry
                                     02E4  1678 ;       <1:0>=1 specifying service on the interrupt stack will cause
                                     02E4  1679 ;       the processor to HALT (code = B).
                                     02E4  1680 ;
                                     02E4  1681 ; Subtest steps:
                                     02E4  1682 ;
                                     02E4  1683 ;       SUBTEST CHME instruction with vector service on IS
                                     02E4  1684 ;       :  Initialize Flags
                                     02E4  1685 ;       :  Get interval timer vector to prevent outside interrupts
                                     02E4  1686 ;       :  Get CHME vector specifying service on IS
                                     02E4  1687 ;       :  Setup code to set done flag at loc 100
                                     02E4  1688 ;       :  Setup code to jump to RENTRY point in next loc
                                     02E4  1689 ;       :  Print restart instructions to operator
                                     02E4  1690 ;       :  Set ready to execute flag
                                     02E4  1691 ;       :  Execute CHME to cause processor to HALT
                                     02E4  1692 ;       :  (this should cause a CHME service on interrupt stack HALT)
                                     02E4  1693 ;       :  IF CHME does not caues an exception
                                     02E4  1694 ;       :  :  THEN
                                     02E4  1695 ;       :  :  Set exception error flag
                                     02E4  1696 ;       :  ENDIF
                                     02E4  1697 ;       :  RENTRY point (restart enters here from exception)
                                     02E4  1698 ;       :  IF exception error flag set
                                     02E4  1699 ;       :  :  THEN
                                     02E4  1700 ;       :  :  Report CHME exception error
                                     02E4  1701 ;       :  :  Exit test
                                     02E4  1702 ;       :  ENDIF
                                     02E4  1703 ;       :  IF CHME instruction executed flag set
                                     02E4  1704 ;       :  :  THEN
                                     02E4  1705 ;       :  :  Report change mode error
                                     02E4  1706 ;       :  :  Exit test
                                     02E4  1707 ;       :  ENDIF
                                     02E4  1708 ;       :  IF done flag not set
                                     02E4  1709 ;       :  :  THEN
                                     02E4  1710 ;       :  :  Report done error
                                     02E4  1711 ;       :  :  Exit test
                                     02E4  1712 ;       :  ENDIF
                                     02E4  1713 ;       :  Read PSL
                                     02E4  1714 ;       :  IF IS not clear
                                     02E4  1715 ;       :  :  THEN
                                     02E4  1716 ;       :  :  Report incorrect stack error in clean up
                                     02E4  1717 ;       :  :  Exit test
                                     02E4  1718 ;       :  ENDIF
                                     02E4  1719 ;       :  Release interval timer vector
                                     02E4  1720 ;       :  Release CHME vector
                                     02E4  1721 ;       ENDSUBTEST CHME instruction with
                                     02E4  1722 ;
                                     02E4  1723 ;-
                                     02E4  1724         $DS_BGNSUB
                                     02E4       T5_S3::
     00010030 9F    0000006C'EF   FA 02E4                   CALLG   $S$, a#DS$BGNSUB
                    00000C2A'EF   16 02EF  1725         JSB     CLEAR_BUFFER            ; clear data buffers
```

```
                 0000003C'EF   D4  02F5  1726          CLRL    L_FLAGS                   ; initialize flags
                                   02FB  1727 1$:      $DS_SETVEC_S    VECTOR=#SCB$L_CHME,SRVADR=CHMX_SERVICE,CODE=#1
                          01   DD  02FB                        PUSHL   #1
                 00000CC4'EF   DF  02FD                        PUSHAL  CHMX_SERVICE
                 00000044 8F   DD  0303                        PUSHL   #SCB$L_CHME
           00010160 9F    03   FB  0309                        CALLS   #3, @#DS$SETVEC
                                   0310  1728
      00000100'EF 00000040'EF  7D  0310  1729 3$:      MOVQ    JUMP,RESTART              ; setup code to renter after halt
      00000108'EF 00000048'EF  7D  031B  1730          MOVQ    JUMP+8,RESTART+8          ; load rest of rentry code
      00000004'EF 000003A0'EF  DE  0326  1731          MOVAL   RENTRY17,RENTRY_POINT     ; load rentry address
                                   0331  1732 4$:      $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#6$,P1=#CODE_0B
                          0B   DD  0331                        PUSHL   #CODE_0B
                 00000397'8F   DD  0333                        PUSHL   #6$
                 000001DC'EF   9F  0339                        PUSHAB  T_INSTRUCTIONS
           000100F0 9F    03   FB  033F                        CALLS   #$$N, @#DS$PRINTF
                                   0346  1733          $DS_PRINTF_S    FORMAT=T_CONTINUE
                 00000212'EF   9F  0346                        PUSHAB  T_CONTINUE
           000100F0 9F    01   FB  034C                        CALLS   #$$N, @#DS$PRINTF
      00000020'EF 00000000'EF  D0  0353  1734 5$:      MOVL    L_ISP,L_CUR_ISP           ; save current IS SP
      00000000'EF 00000000'EF  D0  035E  1735          MOVL    L_KSP,L_EXP_SP            ; load expected SP
      00000000'EF 00000000'EF  D0  0369  1736          MOVL    L_KSP,L_EXP_KSP           ; load expected KSP
      00000000'EF 00000000'EF  D0  0374  1737          MOVL    L_ISP,L_EXP_ISP           ; load expected ISP
      00000000'EF 0000097F'EF  D0  037F  1738          MOVL    DATA_1,L_EXP_PSL          ; load expected PSL
                 00000C42'EF   16  038A  1739          JSB     LOAD_GPR                  ; load GPR's with data
           0000003C'EF    01   C8  0390  1740          BISL2   #M_READY,L_FLAGS          ; set ready to execute flag
                          00   BD  0397  1741 6$:      CHME    #0                        ; execute change mode instruction
           0000003C'EF    04   C8  0399  1742          BISL2   #M_EXECUTE_ERR,L_FLAGS    ; if no exception occurs will get here
                                   03A0  1743 RENTRY17:                                  ; will renter here when completed
        1C 00000000'EF    1A   E1  03A0  1744          BBC     #PSL$V_IS,L_ACT_PSL,9$    ; br = not on IS as expected
           5E    00000020'EF   D0  03A8  1745          MOVL    L_CUR_ISP,SP              ; reset IS SP
           6E    000003B7'8F   D0  03AF  1746          MOVL    #7$,(SP)                  ; setup return address
                               02  03B6  1747          REI                               ; get off interrupt stack
                          0B   11  03B7  1748 7$:      BRB     9$                        ; skip IPR load
      00000000'8F 00000020'EF  DA  03B9  1749 8$:      MTPR    L_CUR_ISP,#PR$_ISP        ; load IS SP into IPR
                 00000CE5'EF   16  03C4  1750 9$:      JSB     ERROR_CHECK               ; check for errors (GPR's,AP,FP,PSL)
        2C 0000003C'EF    0C   E1  03CA  1751          BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                                   03D2  1752          $DS_ERRHARD_S   UNIT=KA_UNIT_NO, PRLINK=ERR_MESSAGE,P1=#T_CHMESCB_ERR
                 0000052B'8F   DD  03D2                        PUSHL   #T_CHMESCB_ERR
                 0000099B'EF   DF  03D8                        PUSHAL  ERR_MESSAGE
                          00   DD  03DE                        PUSHL   #0
                 00000000'EF   DD  03E0                        PUSHL   KA_UNIT_NO
                          01   DD  03E6                        PUSHL   #$ER
                                   03E8           ;;; TEST 5, SUBTEST 3, ERROR 1
           000100D0 9F    05   FB  03E8                        CALLS   #$$M, @#DS$ERRHARD
        07 0000003C'EF    08   E1  03EF  1753          BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                                   03F7  1754          $DS_ABORT
           00010020 9F    6C   FA  03F7                        CALLG   (AP), @#DS$ABORT
                                   03FE  1755 10$:     $DS_CLRVEC_S    VECTOR=#SCB$L_CHME
                 00000044 8F   DD  03FE                        PUSHL   #SCB$L_CHME
           00010168 9F    01   FB  0404                        CALLS   #1, @#DS$CLRVEC
                                   040B  1756          $DS_ENDSUB
                                   040B           TS_S3_X:
      00010038 9F 0000006C'EF  FA  040B                        CALLG   $S$, @#DS$ENDSUB
                                   0416  1757          $DS_PAGE
```

```
                                     0416                .SBTTL  SUBTEST 5.4:  CHMK inst. with vector service on IS
                                     0416       ;+
                                     0416       ; Subtest description:
                                     0416       ;
                                     0416       ;       This subtest checks that a CHMK instruction with the SCB vector entry
                                     0416       ;       <1:0>=1 specifying service on the interrupt stack will cause
                                     0416       ;       the processor to HALT (code = B).
                                     0416       ;
                                     0416       ; Subtest steps:
                                     0416       ;
                                     0416       ;       SUBTEST CHMK instruction with vector service on IS
                                     0416       ;       :  Initialize Flags
                                     0416       ;       :  Get interval timer vector to prevent outside interrupts
                                     0416       ;       :  Get CHMK vector specifying service on IS
                                     0416       ;       :  Setup code to set done flag at loc 100
                                     0416       ;       :  Setup code to jump to RENTRY point in next loc
                                     0416       ;       :  Print restart instructions to operator
                                     0416       ;       :  Set ready to execute flag
                                     0416       ;       :  Execute CHMK to cause processor to HALT
                                     0416       ;       :  (this should cause a CHMK service on interrupt stack HALT)
                                     0416       ;       :  IF CHMK does not caues an exception
                                     0416       ;       :  :  THEN
                                     0416       ;       :  :  Set exception error flag
                                     0416       ;       :  ENDIF
                                     0416       ;       :  RENTRY point (restart enters here from exception)
                                     0416       ;       :  IF exception error flag set
                                     0416       ;       :  :  THEN
                                     0416       ;       :  :  Report CHMK exception error
                                     0416       ;       :  :  Exit test
                                     0416       ;       :  ENDIF
                                     0416       ;       :  IF CHMK instruction executed flag set
                                     0416       ;       :  :  THEN
                                     0416       ;       :  :  Report change mode error
                                     0416       ;       :  :  Exit test
                                     0416       ;       :  ENDIF
                                     0416       ;       :  IF done flag not set
                                     0416       ;       :  :  THEN
                                     0416       ;       :  :  Report done error
                                     0416       ;       :  :  Exit test
                                     0416       ;       :  ENDIF
                                     0416       ;       :  Read PSL
                                     0416       ;       :  IF IS not clear
                                     0416       ;       :  :  THEN
                                     0416       ;       :  :  Report incorrect stack error in clean up
                                     0416       ;       :  :  Exit test
                                     0416       ;       :  ENDIF
                                     0416       ;       :  Release interval timer vector
                                     0416       ;       :  Release CHMK vector
                                     0416       ;       ENDSUBTEST CHMK instruction with
                                     0416       ;
                                     0416       ;-
                                     0416               $DS_BGNSUB
                                     0416       T5_S4::
      00010030 9F     00000078'EF  FA  0416                       CALLG   $S$, @#DS$BGNSUB
                      00000C2A'EF  16  0421             JSB     CLEAR_BUFFER            ; clear data buffers
```

```
                    0000003C'EF   D4  0427  1812          CLRL     L_FLAGS                  ; initialize flags
                                      042D  1813 1$:      $DS_SETVEC_S     VECTOR=#SCB$L_CHMK,SRVADR=CHMX_SERVICE,CODE=#0
                             00   DD  042D                        PUSHL   #0
                    00000CC4'EF   DF  042F                        PUSHAL  CHMX_SERVICE
                    00000040 8F   DD  0435                        PUSHL   #SCB$L_CHMK
               00010160 9F   03   FB  043B                        CALLS   #3, @#DS$SETVEC
                                      0442  1814
     00000100'EF    00000040'EF   7D  0442  1815 3$:      MOVQ     JUMP,RESTART             ; setup code to renter after halt
     00000108'EF    00000048'EF   7D  044D  1816          MOVQ     JUMP+8,RESTART+8         ; load rest of rentry code
     00000004'EF    000004EF'EF   DE  0458  1817          MOVAL    RENTRY18,RENTRY_POINT    ; load rentry address
                                      0463  1818 4$:      $DS_PRINTF_S     FORMAT=T_INSTRUCTIONS,P0=#6$,P1=#CODE_0B
                             0B   DD  0463                        PUSHL   #CODE_0B
                    000004E6'8F   DD  0465                        PUSHL   #6$
                    000001DC'EF   9F  046B                        PUSHAB  T_INSTRUCTIONS
               000100F0 9F   03   FB  0471                        CALLS   #$$N, @#DS$PRINTF
                                      0478  1819          $DS_PRINTF_S     FORMAT=T_CONTINUE
                    00000212'EF   9F  0478                        PUSHAB  T_CONTINUE
               000100F0 9F   01   FB  047E                        CALLS   #$$N, @#DS$PRINTF
     00000020'EF    00000000'EF   D0  0485  1820 5$:      MOVL     L_ISP,L_CUR_ISP          ; save current IS SP
     00000000'EF    00000000'EF   D0  0490  1821          MOVL     L_KSP,L_EXP_SP           ; load expected SP
     00000000'EF    00000000'EF   D0  049B  1822          MOVL     L_KSP,L_EXP_KSP          ; load expected KSP
     00000000'EF    00000000'EF   D0  04A6  1823          MOVL     L_ISP,L_EXP_ISP          ; load expected ISP
     00000000'EF    0000097F'EF   D0  04B1  1824          MOVL     DATA_1,L_EXP_PSL         ; load expected PSL
                    00000C42'EF   16  04BC  1825          JSB      LOAD_GPR                 ; load GPR's with data
     00000030'EF    00000000'8F   DB  04C2  1826          MFPR     #PR$_SCBB,L_TEMP         ; load SCB base
     00000030'EF    00000040 8F   C0  04CD  1827          ADDL2    #SCB$L_CHMK,L_TEMP       ; offset of CHMK vector
               00000030'FF   01   C8  04D8  1828          BISL2    #1,@L_TEMP               ; set service on inter. stack
               0000003C'EF   01   C8  04DF  1829          BISL2    #M_READY,L_FLAGS         ; set ready to execute flag
                             00   BC  04E6  1830 6$:      CHMK     #0                       ; execute change mode instruction
               0000003C'EF   04   C8  04E8  1831          BISL2    #M_EXECUTE_ERR,L_FLAGS   ; if no exception occurs will get here
                                      04EF  1832 RENTRY18:                                  ; will renter here when completed
               00000030'FF   01   CA  04EF  1833          BICL2    #1,@L_TEMP               ; set service on inter. stack
            1C 00000000'EF   1A   E1  04F6  1834          BBC      #PSL$V_IS,L_ACT_PSL,9$   ; br = not on IS as expected
               5E    00000020'EF  D0  04FE  1835          MOVL     L_CUR_ISP,SP             ; reset IS SP
               6E    0000050D'8F  D0  0505  1836          MOVL     #7$,(SP)                 ; setup return address
                                  02  050C  1837          REI                               ; get off interrupt stack
                             0B   11  050D  1838 7$:      BRB      9$                       ; skip IPR load
     00000000'8F    00000020'EF   DA  050F  1839 8$:      MTPR     L_CUR_ISP,#PR$_ISP       ; load IS SP into IPR
                    00000CE5'EF   16  051A  1840 9$:      JSB      ERROR_CHECK              ; check for errors (GPR's,AP,FP,PSL)
            2C 0000003C'EF   0C   E1  0520  1841          BBC      #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                                      0528  1842          $DS_ERRHARD_S    UNIT=KA_UNIT_NO, PRLINK=ERR_MESSAGE,P1=#T_CHMKSCB_ERR
                    00000566'8F   DD  0528                        PUSHL   #T_CHMKSCB_ERR
                    0000099B'EF   DF  052E                        PUSHAL  ERR_MESSAGE
                             00   DD  0534                        PUSHL   #0
                    00000000'EF   DD  0536                        PUSHL   KA_UNIT_NO
                             01   DD  053C                        PUSHL   #$ER
                                      053E       ;;; TEST 5, SUBTEST 4, ERROR 1
               000100D0 9F   05   FB  053E                        CALLS   #$$M, @#DS$ERRHARD
            07 0000003C'EF   08   E1  0545  1843          BBC      #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                                      054D  1844          $DS_ABORT
               00010020 9F   6C   FA  054D                        CALLG   (AP), @#DS$ABORT
                                      0554  1845 10$:     $DS_CLRVEC_S     VECTOR=#SCB$L_CHMK
                    00000040 8F   DD  0554                        PUSHL   #SCB$L_CHMK
               00010168 9F   01   FB  055A                        CALLS   #1, @#DS$CLRVEC
                                      0561  1846          $DS_ENDSUB
```

```
                               0561        T5_S4_X:
    00010038 9F    00000078'EF    FA  0561                     CALLG   $S$, a#DS$ENDSUB
                               056C  1847          $DS_PAGE
                               056C  1848
```

```
                                056C                 .SBTTL  SUBTEST 5.5:  Invalid vector HALT
                                056C        ;+
                                056C        ; Subtest description:
                                056C        ;
                                056C        ;        This subtest checks that an XFC instruction with the SCB vector entry
                                056C        ;        <1:0>=3 specifying an invalid vector will cause the processor to HALT
                                056C        ;        (code = 7).
                                056C        ;
                                056C        ; Subtest steps:
                                056C        ;
                                056C        ;        SUBTEST Invalid vector HALT
                                056C        ;        :  Initialize Flags
                                056C        ;        :  Get interval timer vector to prevent outside interrupts
                                056C        ;        :  Get XFC vector bits <1:0> = 3 (invalid vector)
                                056C        ;        :  Setup code to set done flag at loc 100
                                056C        ;        :  Setup code to jump to RENTRY point in next loc
                                056C        ;        :  Print restart instructions to operator
                                056C        ;        :  Set ready to execute flag
                                056C        ;        :  Execute XFC to cause processor to HALT
                                056C        ;        :  (this should cause a invalid vector HALT)
                                056C        ;        :  IF XFC does not caues an exception
                                056C        ;        :  :  THEN
                                056C        ;        :  :  Set exception error flag
                                056C        ;        :  ENDIF
                                056C        ;        :  RENTRY point (restart enters here from exception)
                                056C        ;        :  IF exception error flag set
                                056C        ;        :  :  THEN
                                056C        ;        :  :  Report XFC exception error
                                056C        ;        :  :  Exit test
                                056C        ;        :  ENDIF
                                056C        ;        :  IF XFC instruction executed flag set
                                056C        ;        :  :  THEN
                                056C        ;        :  :  Report XFC error
                                056C        ;        :  :  Exit test
                                056C        ;        :  ENDIF
                                056C        ;        :  IF done flag not set
                                056C        ;        :  :  THEN
                                056C        ;        :  :  Report done error
                                056C        ;        :  :  Exit test
                                056C        ;        :  ENDIF
                                056C        ;        :  Read PSL
                                056C        ;        :  IF IS not clear
                                056C        ;        :  :  THEN
                                056C        ;        :  :  Report incorrect stack error in clean up
                                056C        ;        :  :  Exit test
                                056C        ;        :  ENDIF
                                056C        ;        :  Release interval timer vector
                                056C        ;        :  Release XFC vector
                                056C        ;        ENDSUBTEST Invalid vector HALT
                                056C        ;
                                056C        ;-
                                056C                 $DS_BGNSUB
                                056C        T5_S5::
  00010030 9F    00000084'EF    FA      056C                         CALLG   $S$, a#DS$BGNSUB
                 00000C2A'EF    16      0577                 JSB     CLEAR_BUFFER            ; clear data buffers
```

```
              0000003C'EF    D4  057D  1903          CLRL     L_FLAGS                   ; initialize flags
                                 0583  1904          $DS_SETVEC_S     VECTOR=#SCB$L_OPCCUS,SRVADR=XFC_ERROR,CODE=#3
                       03    DD  0583                         PUSHL    #3
              00000CCC'EF    DF  0585                         PUSHAL   XFC_ERROR
                       14    DD  058B                         PUSHL    #SCB$L_OPCCUS
          00010160 9F  03    FB  058D                         CALLS    #3, @#DS$SETVEC
00000100'EF   00000040'EF    7D  0594  1905 3$:      MOVQ     JUMP,RESTART              ; setup code to renter after halt
00000108'EF   00000048'EF    7D  059F  1906          MOVQ     JUMP+8,RESTART+8          ; load rest of rentry code
00000004'EF   0000062A'EF    DE  05AA  1907          MOVAL    RENTRY19,RENTRY_POINT     ; load rentry address
00000030'EF   00000000'8F    DB  05B5  1908          MFPR     #PR$_SCBB,L_TEMP
          00000030'EF  14    C0  05C0  1909          ADDL2    #^X14,L_TEMP              ; micro code will halt at SCBB+14
                                 05C7  1910 4$:      $DS_PRINTF_S     FORMAT=T_INSTRUCTIONS,P0=L_TEMP,P1=#CODE_07
                       07    DD  05C7                         PUSHL    #CODE_07
              00000030'EF    DD  05C9                         PUSHL    L_TEMP
              000001DC'EF    9F  05CF                         PUSHAB   T_INSTRUCTIONS
          000100F0 9F  03    FB  05D5                         CALLS    #$$N, @#DS$PRINTF
                                 05DC  1911          $DS_PRINTF_S     FORMAT=T_CONTINUE
              00000212'EF    9F  05DC                         PUSHAB   T_CONTINUE
          000100F0 9F  01    FB  05E2                         CALLS    #$$N, @#DS$PRINTF
00000000'EF   00000000'EF    D0  05E9  1912 5$:      MOVL     L_KSP,L_EXP_SP            ; load expected SP
00000000'EF   00000000'EF    D0  05F4  1913          MOVL     L_KSP,L_EXP_KSP           ; load expected KSP
00000000'EF   00000000'EF    D0  05FF  1914          MOVL     L_ISP,L_EXP_ISP           ; load expected ISP
00000000'EF   0000097F'EF    D0  060A  1915          MOVL     DATA_1,L_EXP_PSL          ; load expected PSL
              00000C42'EF    16  0615  1916          JSB      LOAD_GPR                  ; load GPR's with data
          0000003C'EF  01    C8  061B  1917          BISL2    #M_READY,L_FLAGS          ; set ready to execute flag
                             FC  0622  1918 6$:      XFC                                ; execute XFC to cause exception
          0000003C'EF  04    C8  0623  1919          BISL2    #M_EXECUTE_ERR,L_FLAGS    ; if no exception occurs will get here
                                 062A  1920 RENTRY19:
          5E  00000000'EF    D0  062A  1921 9$:      MOVL     L_KSP,SP                  ; restore SP
              00000CE5'EF    16  0631  1922          JSB      ERROR_CHECK               ; check for errors (GPR's,AP,FP,PSL)
       2C 0000003C'EF  0C    E1  0637  1923          BBC      #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                                 063F  1924          $DS_ERRHARD_S    UNIT=KA_UNIT_NO, PRLINK=ERR_MESSAGE,P1=#T_SCB_ERR
              0000042D'8F    DD  063F                         PUSHL    #T_SCB_ERR
              0000099B'EF    DF  0645                         PUSHAL   ERR_MESSAGE
                       00    DD  064B                         PUSHL    #0
              00000000'EF    DD  064D                         PUSHL    KA_UNIT_NO
                       01    DD  0653                         PUSHL    #$ER
                                 0655               ;;; TEST 5, SUBTEST 5, ERROR 1
          000100D0 9F  05    FB  0655                         CALLS    #$$M, @#DS$ERRHARD
       07 0000003C'EF  08    E1  065C  1925          BBC      #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                                 0664  1926          $DS_ABORT
          00010020 9F  6C    FA  0664                         CALLG    (AP), @#DS$ABORT
                                 066B  1927 10$:     $DS_CLRVEC_S     VECTOR=#SCB$L_OPCCUS
                       14    DD  066B                         PUSHL    #SCB$L_OPCCUS
          00010168 9F  01    FB  066D                         CALLS    #1, @#DS$CLRVEC
                                 0674  1928          $DS_ENDSUB
                                 0674               T5_S5_X:
00010038 9F   00000084'EF    FA  0674                         CALLG    $S$, @#DS$ENDSUB
                                 067F  1929          $DS_CLRVEC_S     VECTOR=#SCB$L_TIMER
              000000C0 8F    DD  067F                         PUSHL    #SCB$L_TIMER
          00010168 9F  01    FB  0685                         CALLS    #1, @#DS$CLRVEC
                                 068C  1930          $DS_PAGE
                   50  01    D0  068C                         MOVL     #1, R0             ; NORMAL EXIT
                                 068F               TEST_005_X::
          00010058 9F  6E    FA  068F                         CALLG    (SP), @#DS$BREAK
```

```
                             04  0696                     RET                      ; RETURN TO TEST SEQUENCER
                                 0697  1932
                           00000000                       .PSECT  $TSTCNT, NOEXE, NOWRT, OVR, LONG
                       00000005  0000                     .LONG   $TN
                                 0004  1934         .END
```

```
$$$                            = 00000084 R     05        DS$ASKADR                        00010090
$$ARGS                         = 0000000A                 DS$ASKDATA                       00010080
$$E                            = 00000001                 DS$ASKLGCL                       00010098
$$M                            = 00000005                 DS$ASKSTR                        000100A0
$$N                            = 00000001                 DS$ASKVLD                        00010088
$$S                            = FFFFFFFF                 DS$ATTACH                        000101A8
$$T1                           = 00000000                 DS$BGNSUB                        00010030
$ENV                           = 00000001  G              DS$BRANCH                        000100A8
$ER                            = FFFFFFFF                 DS$BREAK                         00010058
$MO                            = 00000001  G              DS$CANWAIT                       00010070
$S$                            = 00000084 R     05        DS$CHANNEL                       00010180
$ST                            = 00000000                 DS$CKLOOP                        00010040
$TN                            = 00000005                 DS$CLRVEC                        00010168
ALL                            = 00000009  G              DS$CNTRLC                        00010078
A_BUFFER_BEGIN                   ********   X   03        DS$CVTREG                        000100B0
A_HALT_BUFFER                    ********   X   01        DS$ENDPASS                       00010010
A_PSLMNE                         ********   X   01        DS$ENDSUB                        00010038
BASE_004                         00000000 R     03        DS$ERRDEV                        000100C8
BASE_005                         00000000 R     06        DS$ERRHARD                       000100D0
BUFSIZ                         = 00000080                 DS$ERRPREP                       00010108
B_ANSWER                         00000000 R     01        DS$ERRSOFT                       000100D8
CEP_FUNCTIONAL                 = 00000000                 DS$ERRSYS                        000100C0
CEP_REPAIR                     = 00000001                 DS$ESCAPE                        00010050
CHMX_SERVICE                     00000CC4 R     01        DS$FREEDBGSYM                    000101B8
CLEAR_BUFFER                     00000C2A R     01        DS$GETBUF                        00010120
CLEAR_RPB                        00000BF8 R     01        DS$GETMEM                        00010130
CODE_00                        = 00000000                 DS$GETTERM                       000101C8
CODE_02                        = 00000002                 DS$GPHARD                        00010018
CODE_04                        = 00000004                 DS$HELP                          000101B0
CODE_05                        = 00000005                 DS$INITSCB                       00010170
CODE_06                        = 00000006                 DS$INLOOP                        00010048
CODE_07                        = 00000007                 DS$K_ERROR                     = 00000002
CODE_08                        = 00000008                 DS$K_NORMAL                    = 00000001
CODE_0A                        = 0000000A                 DS$K_SEVERE                    = 00000004
CODE_0B                        = 0000000B                 DS$K_SUBSYS                    = 00000066
CODE_0C                        = 0000000C                 DS$K_WARNING                   = 00000000
CODE_0F                        = 0000000F                 DS$LOAD                          00010198
CODE_11                        = 00000011                 DS$LOADPCS                       000101C0
CODE_12                        = 00000012                 DS$MAPDBGBLOCK                   00010118
CODE_13                        = 00000013                 DS$MEMSIZE                       000101D8
CODE_14                        = 00000014                 DS$MMOFF                         00010158
CODE_15                        = 00000015                 DS$MMON                          00010150
CODE_16                        = 00000016                 DS$MOVPHY                        00010148
CODE_FF                        = 000000FF                 DS$MOVVRT                        00010140
DATA_004                         0000002C R     03        DS$PARSE                         000100B8
DATA_005                         00000024 R     06        DS$PRINTB                        000100E0
DATA_1                           0000097F R     01        DS$PRINTF                        000100F0
DATA_2                           00000983 R     01        DS$PRINTS                        000100F8
DATA_3                           00000987 R     01        DS$PRINTSIG                      00010100
DATA_4                           0000098B R     01        DS$PRINTX                        000100E8
DATA_5                           0000098F R     01        DS$PROBE                         000101A0
DATA_6                           00000993 R     01        DS$RELBUF                        00010128
DATA_7                           00000997 R     01        DS$RELMEM                        00010138
DEFAULT                        = 00000000  G              DS$SERVICE_DISPATCHER            000101D0
DS$ABORT                         00010020                 DS$SETIPL                        00010178
```

```
DS$SETMAP                         00010188          DSA$GL_FLAGS                    0000FE00
DS$SETPRIEXV                      00010110          DSA$GL_MSGTYP                   0000FE40
DS$SETVEC                         00010160          DSA$GL_PASSES                   0000FE08
DS$SHOCHAN                        00010190          DSA$GL_PASSNO                   0000FE54
DS$SUMMARY                        00010028          DSA$GL_SECTNO                   0000FE10
DS$WAITMS                         00010060          DSA$GL_SID                      0000FE14
DS$WAITUS                         00010068          DSA$GL_SUBTNO                   0000FE4C
DS$_ARITH                       = 006600D0          DSA$GL_TESTNO                   0000FE50
DS$_ASBE                        = 00660118          DSA$GL_UNITS                    0000FE0C
DS$_BADLINK                     = 006600F0          DSA$GQ_MSGPTR                   0000FE68
DS$_BADTYPE                     = 006600E8          DSA$GT_DEVNAM                   0000FE5C
DS$_BIIC                        = 00660120          DSA$V_OPER                    = 0000000C
DS$_CHME                        = 006600A8          ENV$M_DOMAIN                  = 00000002
DS$_CHMK                        = 006600E0          ENV$M_LEVEL                   = 00000001
DS$_DEVNAME                     = 00660108          ENV$M_SUPER                   = 000003FC
DS$_ERROR                       = 00660002          ENV$S_DOMAIN                  = 00000001
DS$_FHWE                        = 00660068          ENV$S_LEVEL                   = 00000001
DS$_FRAGBUF                     = 00660080          ENV$S_SUPER                   = 00000008
DS$_ICBUSY                      = 006600C8          ENV$V_DOMAIN                  = 00000001
DS$_ICERR                       = 006600C0          ENV$V_LEVEL                   = 00000000
DS$_IHWE                        = 00660060          ENV$V_SUPER                   = 00000002
DS$_ILLCHAR                     = 00660018          ENV$_CPU                      = 00000000
DS$_ILLPAGCNT                   = 00660078          ENV$_FUNCTIONAL               = 00000000
DS$_ILLUNIT                     = 00660100          ENV$_REPAIR                   = 00000001
DS$_INSFMEM                     = 00660050          ENV$_SUPER                    = 00000001
DS$_IPL2HI                      = 006600B8          ENV$_SYSTEM                   = 00000001
DS$_IVADDR                      = 00660040          ERR$_MSGADR                   = 0000000C
DS$_IVVECT                      = 00660038          ERR$_NARGS                    = 0000000A
DS$_KRNLSTK                     = 00660090          ERR$_NUM                      = 00000004
DS$_LOGIC                       = 00660070          ERR$_P1                       = 00000014
DS$_MCHK                        = 00660088          ERR$_P2                       = 00000018
DS$_MMOFF                       = 00660058          ERR$_P3                       = 0000001C
DS$_NEEDUNIT                    = 006600F8          ERR$_P4                       = 00000020
DS$_NODE                        = 00660128          ERR$_P5                       = 00000024
DS$_NOPCS                       = 00660110          ERR$_P6                       = 00000028
DS$_NORMAL                      = 00660001          ERR$_POINTER                  = 00000010
DS$_NOSUPPORT                   = 006600B1          ERR$_UNIT                     = 00000008
DS$_NOTDON                      = 00660030          ERROR_CHECK                     00000CE5 RG      01
DS$_NOTIMP                      = 006600B0          ERR_MESSAGE                     0000099B R       01
DS$_NULLSTR                     = 00660010          GPR_TABLE                       ********   X     01
DS$_OVERFLOW                    = 00660008          HALT                          = 00000004  G
DS$_POWER                       = 00660098          IPR                           = 00000006  G
DS$_PROGERR                     = 00660020          JUMP                            00000040 R       01
DS$_SEVERE                      = 00660004          KA_UNIT_NO                    = 00000000
DS$_TRANSL                      = 006600A0          LOAD_GPR                        00000C42 R       01
DS$_TRUNCATE                    = 00660028          L_ACT_ISP                       ********   X     01
DS$_UNEXPINT                    = 006600D8          L_ACT_KSP                       ********   X     01
DS$_VASFULL                     = 00660048          L_ACT_PSL                       ********   X     01
DS$_WARNING                     = 00660000          L_ACT_SP                        ********   X     01
DSA$AL_APTMAIL                    0000FE00          L_ADDRESS                       0000001C R       01
DSA$AT_APTTXT                     0000FA00          L_BNXM_ADR                    = 00E00000
DSA$GL_APTCOM                     0000FE04          L_COUNT                         00000018 R       01
DSA$GL_DEVLEN                     0000FE58          L_CUR_ISP                       00000020 R       01
DSA$GL_ERRNO                      0000FE44          L_CUR_SCBB                      00000034 R       01
DSA$GL_EVENT                      0000FE48          L_EXP_AP                        ********   X     01
```

```
L_EXP_FP                         ********   X   01       PSL$M_ZBIT                     = 00000004
L_EXP_ISP                        ********   X   01       PSL$S_CURMOD                   = 00000002
L_EXP_KSP                        ********   X   01       PSL$V_CBIT                     = 00000000
L_EXP_MOD                        00000024 R     01       PSL$V_CURMOD                   = 00000018
L_EXP_PSL                        ********   X   01       PSL$V_IS                       = 0000001A
L_EXP_SP                         ********   X   01       PSL$V_NBIT                     = 00000003
L_EXP_STACK                      00000028 R     01       PSL$V_VBIT                     = 00000001
L_FLAGS                          0000003C R     01       PSL$V_ZBIT                     = 00000002
L_ISP                            ********   X   03       READ_GPR                         00000C6A R     01
L_KA_FLAGS                       00000038 R     01       REG_TABLE                        ********   X   01
L_KSP                            ********   X   03       RENTRY1                          00000159 R     03
L_NXM_ADR                      = 00500200                RENTRY10                         000008EC R     03
L_OLDPROT                        0000002C R     01       RENTRY11                         000009AD R     03
L_PAGE0END                       00000014 R     01       RENTRY12                         00000A6C R     03
L_PAGEND                         0000000C R     01       RENTRY13                         00000B2D R     03
L_PAGESTRT                       00000008 R     01       RENTRY15                         0000013C R     06
L_PAGEZERO                       00000010 R     01       RENTRY16                         0000026E R     06
L_TEMP                           00000030 R     01       RENTRY17                         000003A0 R     06
MANUAL                         = 00000001  G             RENTRY18                         000004EF R     06
MCHK                           = 00000005  G             RENTRY19                         0000062A R     06
MODELIST                         ********   X   01       RENTRY2                          00000220 R     03
MSG_004                          00000002 R     03       RENTRY3                          00000363 R     03
MSG_005                          00000002 R     06       RENTRY4                          00000450 R     03
M_BAD_MODE                     = 00000040                RENTRY5                          00000530 R     03
M_BAD_STK                      = 00000080                RENTRY6                          000005EC R     03
M_CHMX_ERR                     = 00000008                RENTRY7                          000006AD R     03
M_DONE                         = 00000002                RENTRY8                          0000076C R     03
M_ERROR_FLAG                   = 00001000                RENTRY9                          0000082D R     03
M_EXECUTE_ERR                  = 00000004                RENTRY_POINT                     00000004 R     01
M_FATAL_ERR                    = 00000100                RESTART                        = 00000100
M_GPR_ERR                      = 00000400                SCB$L_ACCESS                     00000020
M_ISP_ERR                      = 00004000                SCB$L_ARITH                      00000034
M_ISTK                         = 00010000                SCB$L_BREAK                      0000002C
M_KSP_ERR                      = 00002000                SCB$L_CHME                       00000044
M_PSL_ERR                      = 00000200                SCB$L_CHMK                       00000040
M_READY                        = 00000001                SCB$L_CHMS                       00000048
M_SP_ERR                       = 00008000                SCB$L_CHMU                       0000004C
M_TIMER                        = 00000020                SCB$L_COMPAT                     00000030
M_XFC                          = 00000800                SCB$L_KNLSTK                     00000008
M_XFC_ERR                      = 00000010                SCB$L_MACHCHK                    00000004
PCS                            = 00000008  G             SCB$L_OPCCUS                     00000014
POWER_FAIL                     = 00000003  G             SCB$L_OPCDEC                     00000010
PR$_ICCS                         ********   X   01       SCB$L_POWER                      0000000C
PR$_ISP                          ********   X   01       SCB$L_RADRMOD                    0000001C
PR$_KSP                          ********   X   01       SCB$L_ROPRAND                    00000018
PR$_SCBB                         ********   X   06       SCB$L_RXDB                       000000F8
PRT$C_KR                         ********   X   03       SCB$L_SFTLVL1                    00000084
PRT$C_KW                         ********   X   03       SCB$L_SFTLVL10                   000000A8
PSL$K_EXEC                     = 00000001                SCB$L_SFTLVL11                   000000AC
PSL$K_KERNEL                   = 00000000                SCB$L_SFTLVL12                   000000B0
PSL$K_SUPER                    = 00000002                SCB$L_SFTLVL13                   000000B4
PSL$K_USER                     = 00000003                SCB$L_SFTLVL14                   000000B8
PSL$M_CBIT                     = 00000001                SCB$L_SFTLVL15                   000000BC
PSL$M_NBIT                     = 00000008                SCB$L_SFTLVL2                    00000088
PSL$M_VBIT                     = 00000002                SCB$L_SFTLVL3                    0000008C
```

```
SCB$L_SFTLVL4                   00000090                T4_S2_X                         0000040E R     03
SCB$L_SFTLVL5                   00000094                T4_S3                           00000419 RG    03
SCB$L_SFTLVL6                   00000098                T4_S3_X                         0000059A R     03
SCB$L_SFTLVL7                   0000009C                T4_S4                           000005A5 RG    03
SCB$L_SFTLVL8                   000000A0                T4_S4_X                         0000071D R     03
SCB$L_SFTLVL9                   000000A4                T4_S5                           00000728 RG    03
SCB$L_TBIT                      00000028                T4_S5_X                         0000089D R     03
SCB$L_TIMER                     000000C0                T4_S6                           000008A8 RG    03
SCB$L_TRANSL                    00000024                T4_S6_X                         00000A1D R     03
SCB$L_TXDB                      000000FC                T4_S7                           00000A28 RG    03
SCB$L_ZERO                      00000000                T4_S7_X                         00000B9D R     03
SEP_FUNCTIONAL                = 00000002                T5_S1                           00000080 RG    06
SEP_REPAIR                    = 00000003                T5_S1_X                         000001A7 R     06
SETUP_RPB                       00000C00 R     01       T5_S2                           000001B2 RG    06
SIZ...                        = 00000001                T5_S2_X                         000002D9 R     06
SYS$ALLOC                       00010238                T5_S3                           000002E4 RG    06
SYS$ASCTIM                      00010248                T5_S3_X                         0000040B R     06
SYS$ASSIGN                      00010250                T5_S4                           00000416 RG    06
SYS$BINTIM                      00010258                T5_S4_X                         00000561 R     06
SYS$CANCEL                      00010260                T5_S5                           0000056C RG    06
SYS$CANTIM                      00010268                T5_S5_X                         00000674 R     06
SYS$CLOSE                       000105B8                TEST_004                        00000030 RG    03
SYS$CLREF                       00010298                TEST_004_X                      00000BB8 RG    03
SYS$CONNECT                     000105C0                TEST_005                        00000028 RG    06
SYS$DALLOC                      000102D8                TEST_005_X                      0000068F RG    06
SYS$DASSGN                      000102E0                TIMER_SERVICE                   00000CB8 R     01
SYS$DISCONNECT                  000105D0                TU58                          = 00000002  G
SYS$FAO                         00010350                T_ACT_PSL                       000008E7 R     01
SYS$FAOL                        00010358                T_BAD_DATA                      000007D4 R     01
SYS$GET                         00010580                T_CHMESCB_ERR                   0000052B R     01
SYS$GETCHN                      000104C8                T_CHME_ERR                      000003DD R     01
SYS$GETTIM                      00010378                T_CHMKSCB_ERR                   00000566 R     01
SYS$LKWSET                      000103A0                T_CHMK_ERR                      00000405 R     01
SYS$NUMTIM                      000103B8                T_CHMSSCB_ERR                   000004F0 R     01
SYS$OPEN                        00010608                T_CHMS_ERR                      000003B5 R     01
SYS$QIO                         000103C8                T_CHMUSCB_ERR                   000004B5 R     01
SYS$QIOW                        00010200                T_CHMU_ERR                      0000038D R     01
SYS$READ                        00010590                T_CHMX_ERR                      0000065B R     01
SYS$READEF                      000103D0                T_CONTINUE                      00000212 R     01
SYS$SETAST                      000103F8                T_DBL_MCHK_ERR                  000000E6 R     01
SYS$SETEF                       00010400                T_EXECUTE                       00000601 R     01
SYS$SETIMR                      00010420                T_EXITING                       000001C7 R     01
SYS$SETPRI                      00010428                T_EXP_PSL                       000008C2 R     01
SYS$SETPRT                      00010430   G            T_GOOD_DATA                     000007FB R     01
SYS$SETRWM                      00010438                T_GPR                           00000929 R     01
SYS$SETSWM                      00010448                T_GPR_ERR                       000007AB R     01
SYS$ULKPAG                      00010460                T_HALT_ERR                      000002F1 R     01
SYS$ULWSET                      00010468                T_HALT_MESS                     000002C1 R     01
SYS$UNWIND                      00010470                T_INSTRUCTIONS                  000001DC R     01
SYS$WAITFR                      00010478                T_INVLDIS_ERR                   00000315 R     01
SYS$WFLAND                      00010488                T_ISP_ERR1                      00000715 R     01
SYS$WFLOR                       00010490                T_ISP_ERR2                      00000743 R     01
T4_S1                           00000088 RG    03       T_KSP_ERR1                      000006B9 R     01
T4_S1_X                         000001A0 R     03       T_KSP_ERR2                      000006E7 R     01
T4_S2                           000001AB RG    03       T_NOT_DONE                      000005D3 R     01
```

```
T_NOT_ON_IS                       0000089B R      01
T_NOT_ON_KS                       00000877 R      01
T_NOT_READY                       000005A1 R      01
T_NOXOR                           0000096B R      01
T_NO_OPER                         0000004D R      01
T_NXMIS_ERR                       00000350 R      01
T_NXM_WCS_ERR                     00000124 R      01
T_PAGEOPROT                       00000816 R      01
T_PAGEPROT                        00000844 R      01
T_PSL                             0000090C R      01
T_REST_OFF                        00000197 R      01
T_SCBBNXM_ERR                     0000015B R      01
T_SCB_ERR                         0000042D R      01
T_SP_ERR1                         00000771 R      01
T_SP_ERR2                         0000079F R      01
T_TIMER                           00000691 R      01
T_WARNING                         00000262 R      01
T_WCS_ERR                         00000463 R      01
T_XFC                             00000626 R      01
T_XOR                             00000951 R      01
V_BAD_MODE                      = 00000006
V_BAD_STK                       = 00000007
V_CHMX_ERR                      = 00000003
V_DONE                          = 00000001
V_ERROR_FLAG                    = 0000000C
V_EXECUTE_ERR                   = 00000002
V_FATAL_ERR                     = 00000008
V_GPR_ERR                       = 0000000A
V_ISP_ERR                       = 0000000E
V_ISTK                          = 00000010
V_KSP_ERR                       = 0000000D
V_PSL_ERR                       = 00000009
V_READY                         = 00000000
V_SP_ERR                        = 0000000F
V_TIMER                         = 00000005
V_USR_WCS                       = 00000019
V_XFC                           = 0000000B
V_XFC_ERR                       = 00000004
WCS                             = 00000007  G
XFC_ERROR                         00000CCC R      01
XFC_SERVICE                       00000CD8 R      01

                                                +-----------------+
                                                ! Psect synopsis !
                                                +-----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (   0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
ECKAXD                          00000E0D  (  3597.)  01 (   1.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC LONG
$ABS$                           00010610  (67088.)  02 (   2.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
TEST_004                        00000BC0  (  3008.)  03 (   3.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD   NOWRT NOVEC PAGE
DISPATCH                        00000030  (    48.)  04 (   4.)  NOPIC    USR    CON    REL    LCL NOSHR NOEXE   RD   NOWRT NOVEC LONG
ARGLIST                         00000090  (   144.)  05 (   5.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD   NOWRT NOVEC LONG
TEST_005                        00000697  (  1687.)  06 (   6.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD   NOWRT NOVEC PAGE
```

```
$TSTCNT                         00000004 (     4.)  07 (  7.)  NOPIC    USR    OVR    REL    LCL NOSHR NOEXE    RD   NOWRT NOVEC LONG
```

```
                                          +-------------------------+
                                          ! Symbol Cross Reference !
                                          +-------------------------+

SYMBOL                        VALUE       DEFINITION       REFERENCES...
-- --                         -----       ----------       -------------
$$$                           =00000084-R 1901     (20)     1057    (11)      1177    (12)      1295    (13)      1414    (14)
                                                            1552    (16)      1638    (17)      1724    (18)      1810    (19)
                                                            1901    (20)      670     (8)       794     (9)       934     (10)
$$ARGS                        =0000000A   90       (2)      90      (2)
$$E                           =00000001   1901     (20)     1103    (11)      1223    (12)      1341    (13)      1460    (14)
                                                            1584    (16)      1670    (17)      1756    (18)      1846    (19)
                                                            1928    (20)      704     (8)       867     (9)       988     (10)
$$M                           =00000005   1924     (20)     1070    (11)      1098    (11)      1190    (12)      1218    (12)
                                                            1308    (13)      1336    (13)      1427    (14)      1455    (14)
                                                            1486    (15)      1580    (16)      1666    (17)      1752    (18)
                                                            1842    (19)      1924    (20)      611     (7)       699     (8)
                                                            802     (9)       810     (9)       829     (9)       857     (9)
                                                            863     (9)       945     (10)      983     (10)
$$N                           -00000001   1924     (20)     1070    (11)      1078    (11)    #-1079    (11)      1098    (11)
                                                            1190    (12)      1198    (12)    #-1199    (12)      1218    (12)
                                                            1308    (13)      1316    (13)    #-1317    (13)      1336    (13)
                                                            1427    (14)      1435    (14)    #-1436    (14)      1455    (14)
                                                          #-1488    (15)    #-1491    (15)      1560    (16)    #-1561    (16)
                                                            1580    (16)      1646    (17)    #-1647    (17)      1666    (17)
                                                            1732    (18)    #-1733    (18)      1752    (18)      1818    (19)
                                                          #-1819    (19)      1842    (19)      1910    (20)    #-1911    (20)
                                                            1924    (20)    #-370     (5)     #-371     (5)     #-373     (5)
                                                          #-375     (5)     #-377     (5)     #-379     (5)     #-381     (5)
                                                            384     (5)       386     (5)       389     (5)       391     (5)
                                                            393     (5)       395     (5)       401     (5)       405     (5)
                                                            410     (5)     #-412     (5)       420     (5)       422     (5)
                                                          #-425     (5)     #-613     (7)     #-615     (7)       680     (8)
                                                          #-681     (8)       699     (8)       802     (9)       810     (9)
                                                            821     (9)     #-822     (9)       829     (9)       857     (9)
                                                            863     (9)       945     (10)      95      (2)       957     (10)
                                                          #-958     (10)      983     (10)
$$S                           =FFFFFFFF   1931     (20)     1057    (11)      1107    (12)      1177    (12)      1226    (13)
                                                            1295    (13)      1344    (14)      1414    (14)      1467    (15)
                                                            1486    (15)      1501    (16)      1552    (16)      1587    (17)
                                                            1638    (17)      1673    (18)      1724    (18)      1759    (19)
                                                            1810    (19)      1850    (20)      1901    (20)      598     (7)
                                                            611     (7)       623     (8)       670     (8)       707     (9)
                                                            794     (9)       870     (10)      934     (10)      991     (11)
$$T1                          =00000000   861      (9)      800     (9)       827     (9)       861     (9)       90      (2)
$ENV                          =00000001   73       (2)
$ER                           =FFFFFFFF   1931     (20)     1057    (11)    #-1070    (11)    #-1098    (11)      1177    (12)
                                                          #-1190    (12)    #-1218    (12)      1295    (13)    #-1308    (13)
                                                          #-1336    (13)      1414    (14)    #-1427    (14)    #-1455    (14)
                                                            1552    (16)    #-1580    (16)      1638    (17)    #-1666    (17)
                                                            1724    (18)    #-1752    (18)      1810    (19)    #-1842    (19)
                                                            1901    (20)    #-1924    (20)      670     (8)     #-699     (8)
                                                            794     (9)     #-802     (9)     #-810     (9)     #-829     (9)
                                                          #-857     (9)     #-863     (9)       934     (10)    #-945     (10)
```

```
                                                           #-983    (10)
$MO                            =00000001     1933   (20)    1933    (20)
$S$                            =00000084-R   1901   (20)    1057    (11)    1103    (11)    1177    (12)    1223    (12)
                                                            1295    (13)    1341    (13)    1414    (14)    1460    (14)
                                                            1552    (16)    1584    (16)    1638    (17)    1670    (17)
                                                            1724    (18)    1756    (18)    1810    (19)    1846    (19)
                                                            1901    (20)    1928    (20)     670    (8)      704    (8)
                                                             794    (9)      867    (9)      934    (10)     988    (10)
$ST                            =00000000     1931   (20)    1057    (11)    1070    (11)    1098    (11)    1103    (11)
                                                            1107    (12)    1177    (12)    1190    (12)    1218    (12)
                                                            1223    (12)    1226    (13)    1295    (13)    1308    (13)
                                                            1336    (13)    1341    (13)    1344    (14)    1414    (14)
                                                            1427    (14)    1455    (14)    1460    (14)    1462    (14)
                                                            1486    (15)    1501    (16)    1552    (16)    1580    (16)
                                                            1584    (16)    1587    (17)    1638    (17)    1666    (17)
                                                            1670    (17)    1673    (18)    1724    (18)    1752    (18)
                                                            1756    (18)    1759    (19)    1810    (19)    1842    (19)
                                                            1846    (19)    1850    (20)    1901    (20)    1924    (20)
                                                            1928    (20)    1931    (20)     611    (7)      623    (8)
                                                             670    (8)      699    (8)      704    (8)      707    (9)
                                                             794    (9)      802    (9)      810    (9)      829    (9)
                                                             857    (9)      863    (9)      867    (9)      870    (10)
                                                             934    (10)     945    (10)     983    (10)     988    (10)
                                                             991    (11)
$TN                            =00000005     1933   (20)    1070    (11)    1098    (11)    1107    (12)    1190    (12)
                                                            1218    (12)    1226    (13)    1308    (13)    1336    (13)
                                                            1344    (14)    1427    (14)    1455    (14)    1462    (14)
                                                            1467    (15)    1486    (15)    1501    (16)    1580    (16)
                                                            1587    (17)    1666    (17)    1673    (18)    1752    (18)
                                                            1759    (19)    1842    (19)    1850    (20)    1924    (20)
                                                            1931    (20)    1933    (20)     598    (7)      611    (7)
                                                             623    (8)      699    (8)      707    (9)      810    (9)
                                                             857    (9)      870    (10)     945    (10)     983    (10)
                                                             991    (11)
ALL                            =00000009     95     (2)     1486    (15)     611    (7)
A BUFFER BEGIN                  00000000-XR                #-823    (9)
A_HALT_BUFFER                   00000000-XR                #-414    (5)      461    (6)    #-488    (6)    #-489    (6)
                                                           #-550    (6)
A_PSLMNE                        00000000-XR                  400    (5)      404    (5)      409    (5)
BASE_004                        00000000-R   598    (7)      611    (7)
BASE_005                        00000000-R   1467   (15)    1486    (15)
BIT...                         =00000003     92     (2)       73    (2)       89    (2)       92    (2)
BUFSIZ                         =00000080     175    (2)    #-368    (5)    #-400    (5)    #-404    (5)    #-409    (5)
B_ANSWER                        00000000-R   98     (2)
CEP_FUNCTIONAL                 =00000000     73     (2)
CEP_REPAIR                     =00000001     73     (2)       73    (2)
CHMX_SERVICE                    00000CC4-R   519    (6)     1062    (11)    1182    (12)    1300    (13)    1419    (14)
                                                            1555    (16)    1641    (17)    1727    (18)    1813    (19)
CLEAR_BUFFER                    00000C2A-R   460    (6)     1059    (11)    1179    (12)    1297    (13)    1416    (14)
                                                            1496    (15)    1553    (16)    1639    (17)    1725    (18)
                                                            1811    (19)    1902    (20)     618    (7)      672    (8)
                                                             796    (9)      936    (10)
CLEAR_RPB                       00000BF8-R   435    (6)     1058    (11)    1178    (12)    1296    (13)    1415    (14)
                                                             671    (8)      795    (9)      935    (10)
CODE_00                        -00000000     156    (2)
```

```
CODE_02                       =00000002    157    (2)
CODE_04                       =00000004    158    (2)    #-821   (9)    #-957   (10)
CODE_05                       =00000005    159    (2)
CODE_06                       =00000006    160    (2)    #-680   (8)
CODE_07                       =00000007    161    (2)    #-1910  (20)
CODE_08                       =00000008    162    (2)
CODE_0A                       =0000000A    163    (2)    #-1078  (11)   #-1198  (12)   #-1316  (13)   #-1435  (14)
CODE_0B                       =0000000B    164    (2)    #-1560  (16)   #-1646  (17)   #-1732  (18)   #-1818  (19)
CODE_0C                       =0000000C    165    (2)
CODE_0F                       =0000000F    166    (2)
CODE_11                       =00000011    167    (2)
CODE_12                       =00000012    168    (2)
CODE_13                       =00000013    169    (2)
CODE_14                       =00000014    170    (2)
CODE_15                       =00000015    171    (2)
CODE_16                       =00000016    172    (2)
CODE_FF                       =000000FF    173    (2)
DATA_004                       0000002C-R  611    (7)      611   (7)
DATA_005                       00000024-R  1486   (15)     1486  (15)
DATA_1                         0000097F-R  350    (5)    #-1566  (16)   #-1652  (17)   #-1738  (18)   #-1824  (19)
                                                         #-1915  (20)   #-685   (8)    #-688   (8)
DATA_2                         00000983-R  352    (5)    #-835   (9)
DATA_3                         00000987-R  354    (5)    #-962   (10)
DATA_4                         0000098B-R  356    (5)    #-1084  (11)   #-1204  (12)   #-1322  (13)   #-1441  (14)
DATA_5                         0000098F-R  358    (5)
DATA_6                         00000993-R  360    (5)
DATA_7                         00000997-R  362    (5)
DEFAULT                       =00000000    95     (2)
DS$ABORT                       00010020                    1071  (11)     1100  (11)     1191  (12)     1220  (12)
                                                           1309  (13)     1338  (13)     1428  (14)     1457  (14)
                                                           1489  (15)     1582  (16)     1668  (17)     1754  (18)
                                                           1844  (19)     1926  (20)     614   (7)      701   (8)
                                                           703   (8)      803   (9)      811   (9)      830   (9)
                                                           858   (9)      860   (9)      864   (9)      946   (10)
                                                           985   (10)
DS$BGNSUB                      00010030                    1057  (11)     1177  (12)     1295  (13)     1414  (14)
                                                           1552  (16)     1638  (17)     1724  (18)     1810  (19)
                                                           1901  (20)     670   (8)      794   (9)      934   (10)
DS$BREAK                       00010058                    1462  (14)     1931  (20)
DS$CLRVEC                      00010168                    1101  (11)     1102  (11)     1221  (12)     1222  (12)
                                                           1339  (13)     1340  (13)     1458  (14)     1459  (14)
                                                           1461  (14)     1583  (16)     1669  (17)     1755  (18)
                                                           1845  (19)     1927  (20)     1929  (20)     812   (9)
                                                           866   (9)      947   (10)     987   (10)
DS$CVTREG                      000100B0                    400   (5)      404   (5)      409   (5)
DS$ENDSUB                      00010038                    1103  (11)     1223  (12)     1341  (13)     1460  (14)
                                                           1584  (16)     1670  (17)     1756  (18)     1846  (19)
                                                           1928  (20)     704   (8)      867   (9)      988   (10)
DS$ERRHARD                     000100D0                    1070  (11)     1098  (11)     1190  (12)     1218  (12)
                                                           1308  (13)     1336  (13)     1427  (14)     1455  (14)
                                                           1580  (16)     1666  (17)     1752  (18)     1842  (19)
                                                           1924  (20)     699   (8)      810   (9)      857   (9)
                                                           945   (10)     983   (10)
DS$ERRSYS                      000100C0                    802   (9)      829   (9)      863   (9)
DS$K_ERROR                    =00000002    89     (2)
```

```
                                                                J 15
ZZ-ECKAX-5.2    Cross reference                                25-FEB-1986      Fiche 1  Frame J15        Sequence 191
ECKAX4                          VAX-11/750 Specific CPU Cluster Exercise 25-FEB-1986 13:15:31  VAX/VMS Macro V04-00          Page  66
Cross reference                                                         25-FEB-1986 13:14:01  [TAYLOR.ECKAX]ECKAX4.MAR;4         (20)

DS$K_NORMAL                    =00000001     89     (2)
DS$K_SEVERE                    =00000004     89     (2)
DS$K_SUBSYS                    =00000066     89     (2)      89     (2)
DS$K_WARNING                   =00000000     89     (2)
DS$MMOFF                        00010158                     865    (9)      986    (10)
DS$MMON                         00010150                     799    (9)      938    (10)
DS$PRINTB                       000100E0                     370    (5)      371    (5)      373    (5)      375    (5)
                                                             377    (5)      379    (5)      381    (5)      384    (5)
                                                             389    (5)      393    (5)      401    (5)      405    (5)
                                                             412    (5)      420    (5)      425    (5)
DS$PRINTF                       000100F0                     1078   (11)     1079   (11)     1198   (12)     1199   (12)
                                                             1316   (13)     1317   (13)     1435   (14)     1436   (14)
                                                             1488   (15)     1491   (15)     1560   (16)     1561   (16)
                                                             1646   (17)     1647   (17)     1732   (18)     1733   (18)
                                                             1818   (19)     1819   (19)     1910   (20)     1911   (20)
                                                             613    (7)      615    (7)      680    (8)      681    (8)
                                                             821    (9)      822    (9)      957    (10)     958    (10)
DS$PRINTX                       000100E8                     386    (5)      391    (5)      395    (5)      410    (5)
                                                             422    (5)
DS$SETVEC                       00010160                     1061   (11)     1062   (11)     1181   (12)     1182   (12)
                                                             1299   (13)     1300   (13)     1418   (14)     1419   (14)
                                                             1497   (15)     1555   (16)     1641   (17)     1727   (18)
                                                             1813   (19)     1904   (20)     619    (7)      804    (9)
                                                             813    (9)      939    (10)     949    (10)
DS$ ARITH                      =006600D0     89     (2)
DS$ ASBE                       =00660118     89     (2)
DS$_BADLINK                    =006600F0     89     (2)
DS$_BADTYPE                    =006600E8     89     (2)
DS$_BIIC                       =00660120     89     (2)
DS$_CHME                       =006600A8     89     (2)
DS$_CHMK                       =006600E0     89     (2)
DS$_DEVNAME                    =00660108     89     (2)
DS$_ERROR                      =00660002     89     (2)
DS$_FHWE                       =00660068     89     (2)
DS$_FRAGBUF                    =00660080     89     (2)
DS$_ICBUSY                     =006600C8     89     (2)
DS$_ICERR                      =006600C0     89     (2)
DS$_IHWE                       =00660060     89     (2)
DS$_ILLCHAR                    =00660018     89     (2)
DS$_ILLPAGCNT                  =00660078     89     (2)
DS$_ILLUNIT                    =00660100     89     (2)
DS$_INSFMEM                    =00660050     89     (2)
DS$_IPL2HI                     =006600B8     89     (2)
DS$_IVADDR                     =00660040     89     (2)
DS$_IVVECT                     =00660038     89     (2)
DS$_KRNLSTK                    =00660090     89     (2)
DS$_LOGIC                      =00660070     89     (2)
DS$_MCHK                       =00660088     89     (2)
DS$_MMOFF                      =00660058     89     (2)
DS$_NEEDUNIT                   =006600F8     89     (2)
DS$_NODE                       =00660128     89     (2)
DS$_NOPCS                      =00660110     89     (2)
DS$_NORMAL                     =00660001     89     (2)
DS$_NOSUPPORT                  =006600B1     89     (2)
DS$_NOTDON                     =00660030     89     (2)
```

```
DS$_NOTIMP                     =006600B0     89     (2)      89     (2)
DS$_NULLSTR                    =00660010     89     (2)
DS$_OVERFLOW                   =00660008     89     (2)
DS$_POWER                      =00660098     89     (2)
DS$_PROGERR                    =00660020     89     (2)
DS$_SEVERE                     =00660004     89     (2)
DS$_TRANSL                     =006600A0     89     (2)
DS$_TRUNCATE                   =00660028     89     (2)
DS$_UNEXPINT                   =006600D8     89     (2)
DS$_VASFULL                    =00660048     89     (2)
DS$_WARNING                    =00660000     89     (2)      89     (2)
DSA$GL_FLAGS                    0000FE00                     1075   (11)     1195   (12)     1313   (13)     1432   (14)
                                                             1487   (15)     612    (7)      677    (8)      687    (8)
                                                             818    (9)      954    (10)
DSA$V_OPER                     =0000000C                   #-1075   (11)   #-1195   (12)   #-1313   (13)   #-1432   (14)
                                                           #-1487   (15)   #-612    (7)    #-677    (8)    #-687    (8)
                                                           #-818    (9)    #-954    (10)
ENV$M_DOMAIN                   =00000002     73     (2)
ENV$M_LEVEL                    =00000001     73     (2)
ENV$M_SUPER                    =000003FC     73     (2)
ENV$S_DOMAIN                   =00000001     73     (2)
ENV$S_LEVEL                    =00000001     73     (2)
ENV$S_SUPER                    =00000008     73     (2)
ENV$V_DOMAIN                   =00000001     73     (2)      73     (2)
ENV$V_LEVEL                    =00000000     73     (2)      73     (2)
ENV$V_SUPER                    =00000002     73     (2)
ENV$_CPU                       =00000000     73     (2)      73     (2)
ENV$_FUNCTIONAL                =00000000     73     (2)      73     (2)
ENV$_REPAIR                    =00000001     73     (2)      73     (2)
ENV$_SUPER                     =00000001     73     (2)
ENV$_SYSTEM                    =00000001     73     (2)      73     (2)
ERR$_MSGADR                    =0000000C     90     (2)
ERR$_NARGS                     =0000000A     90     (2)
ERR$_NUM                       =00000004     90     (2)
ERR$_P1                        =00000014     90     (2)      371    (5)
ERR$_P2                        =00000018     90     (2)
ERR$_P3                        =0000001C     90     (2)
ERR$_P4                        =00000020     90     (2)
ERR$_P5                        =00000024     90     (2)
ERR$_P6                        =00000028     90     (2)
ERR$_POINTER                   =00000010     90     (2)
ERR$_UNIT                      =00000008     90     (2)
ERROR_CHECK                     00000CE5-R   547    (6)      1096   (11)     1216   (12)     1334   (13)     1453   (14)
                                                             1578   (16)     1664   (17)     1750   (18)     1840   (19)
                                                             1922   (20)     697    (8)      855    (9)      981    (10)
ERR_MESSAGE                     0000099B-R   366    (5)      1098   (11)     1218   (12)     1336   (13)     1455   (14)
                                                             1580   (16)     1666   (17)     1752   (18)     1842   (19)
                                                             1924   (20)     699    (8)      857    (9)      983    (10)
GPR_TABLE                       00000000-XR                  413    (5)      471    (6)      551    (6)
HALT                           =00000004     95     (2)      1486   (15)     611    (7)
IPR                            =00000006     95     (2)
JUMP                            00000040-R   178    (2)    #-1072   (11)   #-1073   (11)   #-1192   (12)   #-1193   (12)
                                                           #-1310   (13)   #-1311   (13)   #-1429   (14)   #-1430   (14)
                                                           #-1557   (16)   #-1558   (16)   #-1643   (17)   #-1644   (17)
                                                           #-1729   (18)   #-1730   (18)   #-1815   (19)   #-1816   (19)
```

```
                                                      #-1905   (20)   #-1906   (20)    #-674   (8)     #-675   (8)
                                                       #-814   (9)     #-815   (9)     #-950   (10)    #-951   (10)
KA_UNIT_NO                    =00000000    117   (2)  #-1580   (16)   #-1666   (17)   #-1752   (18)   #-1842   (19)
                                                      #-1924   (20)
LOAD_GPR                       00000C42-R  470   (6)    1085   (11)     1205   (12)     1323   (13)     1442   (14)
                                                        1567   (16)     1653   (17)     1739   (18)     1825   (19)
                                                        1916   (20)      686   (8)       837   (9)       963   (10)
L_ACT_ISP                      00000000-XR             #-389   (5)     #-391   (5)     #-500   (6)     #-565   (6)
L_ACT_KSP                      00000000-XR             #-384   (5)     #-386   (5)     #-499   (6)     #-561   (6)
L_ACT_PSL                      00000000-XR            #-1068   (11)     1069   (11)     1090   (11)   #-1188   (12)
                                                        1189   (12)     1210   (12)   #-1306   (13)     1307   (13)
                                                        1328   (13)   #-1425   (14)     1426   (14)     1447   (14)
                                                        1572   (16)     1658   (17)     1744   (18)     1834   (19)
                                                       #-404   (5)     #-405   (5)     #-501   (6)       559   (6)
                                                       #-568   (6)       573   (6)       578   (6)     #-808   (9)
                                                         809   (9)       849   (9)     #-943   (10)      944   (10)
                                                         974   (10)
L_ACT_SP                       00000000-XR             #-393   (5)     #-395   (5)     #-498   (6)     #-556   (6)
L_ADDRESS                      0000001C-R  106   (2)   #-823   (9)     #-824   (9)     #-825   (9)     #-826   (9)
                                                       #-832   (9)     #-839   (9)
L_BNXM_ADR                    =00E00000    153   (2)
L_COUNT                        00000018-R  105   (2)   #-451   (6)     #-453   (6)     #-462   (6)     #-464   (6)
L_CUR_ISP                      00000020-R  107   (2)  #-1080   (11)   #-1091   (11)   #-1095   (11)   #-1200   (12)
                                                      #-1211   (12)   #-1215   (12)   #-1318   (13)   #-1329   (13)
                                                      #-1333   (13)   #-1437   (14)   #-1448   (14)   #-1452   (14)
                                                      #-1562   (16)   #-1573   (16)   #-1577   (16)   #-1648   (17)
                                                      #-1659   (17)   #-1663   (17)   #-1734   (18)   #-1745   (18)
                                                      #-1749   (18)   #-1820   (19)   #-1835   (19)   #-1839   (19)
                                                       #-817   (9)     #-850   (9)     #-854   (9)     #-953   (10)
                                                       #-975   (10)    #-979   (10)
L_CUR_SCBB                     00000034-R  112   (2)
L_EXP_AP                       00000000-XR             #-478   (6)     #-689   (8)
L_EXP_FP                       00000000-XR             #-479   (6)
L_EXP_ISP                      00000000-XR            #-1083   (11)   #-1203   (12)   #-1321   (13)   #-1440   (14)
                                                      #-1565   (16)   #-1651   (17)   #-1737   (18)   #-1823   (19)
                                                      #-1914   (20)    #-389   (5)     #-565   (6)     #-684   (8)
                                                       #-834   (9)     #-961   (10)
L_EXP_KSP                      00000000-XR            #-1082   (11)   #-1202   (12)   #-1320   (13)   #-1439   (14)
                                                      #-1564   (16)   #-1650   (17)   #-1736   (18)   #-1822   (19)
                                                      #-1913   (20)    #-384   (5)     #-561   (6)     #-683   (8)
                                                       #-833   (9)     #-960   (10)
L_EXP_MOD                      00000024-R  108   (2)   #-571   (6)     #-573   (6)
L_EXP_PSL                      00000000-XR            #-1084   (11)   #-1204   (12)   #-1322   (13)   #-1441   (14)
                                                      #-1566   (16)   #-1652   (17)   #-1738   (18)   #-1824   (19)
                                                      #-1915   (20)    #-400   (5)     #-401   (5)     #-409   (5)
                                                       #-410   (5)     #-568   (6)       571   (6)       576   (6)
                                                       #-685   (8)     #-688   (8)     #-835   (9)     #-962   (10)
L_EXP_SP                       00000000-XR            #-1081   (11)   #-1201   (12)   #-1319   (13)   #-1438   (14)
                                                      #-1563   (16)   #-1649   (17)   #-1735   (18)   #-1821   (19)
                                                      #-1912   (20)    #-393   (5)     #-556   (6)     #-682   (8)
                                                       #-690   (8)     #-832   (9)     #-959   (10)
L_EXP_STACK                    00000028-R  109   (2)   #-576   (6)     #-578   (6)
L_FLAGS                        0000003C-R  115   (2)  #-1060   (11)   #-1086   (11)   #-1088   (11)     1097   (11)
                                                        1099   (11)   #-1180   (12)   #-1206   (12)   #-1208   (12)
                                                        1217   (12)     1219   (12)   #-1298   (13)   #-1324   (13)
```

```
                                                    #-1326  (13)     1335  (13)     1337  (13)   #-1417  (14)
                                                    #-1443  (14)   #-1445  (14)     1454  (14)     1456  (14)
                                                    #-1554  (16)   #-1568  (16)   #-1570  (16)     1579  (16)
                                                      1581  (16)   #-1640  (17)   #-1654  (17)   #-1656  (17)
                                                      1665  (17)     1667  (17)   #-1726  (18)   #-1740  (18)
                                                    #-1742  (18)     1751  (18)     1753  (18)   #-178   (2)
                                                    #-1812  (19)   #-1829  (19)   #-1831  (19)     1841  (19)
                                                      1843  (19)   #-1903  (20)   #-1917  (20)   #-1919  (20)
                                                      1923  (20)     1925  (20)     372   (5)      374   (5)
                                                      376   (5)      378   (5)      380   (5)      382   (5)
                                                      383   (5)      387   (5)      388   (5)      392   (5)
                                                      396   (5)      411   (5)      424   (5)    #-511   (6)
                                                    #-520   (6)    #-529   (6)    #-538   (6)    #-554   (6)
                                                    #-558   (6)    #-560   (6)    #-563   (6)      564   (6)
                                                    #-567   (6)    #-580   (6)    #-581   (6)      582   (6)
                                                    #-583   (6)      584   (6)    #-585   (6)      586   (6)
                                                    #-587   (6)      588   (6)    #-589   (6)      590   (6)
                                                    #-591   (6)      592   (6)    #-593   (6)    #-673   (8)
                                                    #-691   (8)    #-694   (8)      698   (8)      702   (8)
                                                    #-797   (9)    #-838   (9)    #-845   (9)      856   (9)
                                                      859   (9)    #-937   (10)   #-965   (10)   #-970   (10)
                                                      982   (10)     984   (10)
L_ISP                        00000000-XR            #-1081  (11)   #-1201  (.2)   #-1319  (13)   #-1438  (14)
                                                    #-1495  (15)   #-1562  (16)   #-1565  (16)   #-1648  (17)
                                                    #-1651  (17)   #-1734  (18)   #-1737  (18)   #-1820  (19)
                                                    #-1823  (19)   #-1914  (20)   #-617   (7)    #-684   (8)
L_KA_FLAGS                   00000038-R 113   (2)
L_KSP                        00000000-XR            #-1082  (11)   #-1202  (12)   #-1320  (13)   #-1439  (14)
                                                    #-1494  (15)   #-1563  (16)   #-1564  (16)   #-1649  (17)
                                                    #-1650  (17)   #-1735  (18)   #-1736  (18)   #-1821  (19)
                                                    #-1822  (19)   #-1912  (20)   #-1913  (20)   #-1921  (20)
                                                    #-616   (7)    #-682   (8)    #-696   (8)    #-833   (9)
                                                    #-960   (10)
L_NXM_ADR                   =00500200   152   (2)
L_OLDPROT                    0000002C-R 110   (2)     827   (9)      861   (9)
L_PAGE0END                   00000014-R 103   (2)
L_PAGEND                     0000000C-R 101   (2)   #-825   (9)
L_PAGESTRT                   00000008-R 100   (2)   #-824   (9)      827   (9)      861   (9)
L_PAGEZERO                   00000010-R 102   (2)   #-798   (9)      800   (9)
L_TEMP                       00000030-R 111   (2)   #-1826  (19)   #-1827  (19)   #-1828  (19)   #-1833  (19)
                                                    #-1908  (20)   #-1909  (20)   #-1910  (20)
MANUAL                      =00000001   95    (2)     1486  (15)     611   (7)
MCHK                        =00000005   95    (2)
MODELIST                     00000000-XR              400   (5)      404   (5)      409   (5)
MSG_004                      00000002-R 598   (7)     611   (7)
MSG_005                      00000002-R 1467  (15)    1486  (15)
M_BAD_MODE                  =00000040   131   (2)
M_BAD_STK                   =00000080   133   (2)
M_CHMX_ERR                  =00000008   125   (2)   #-520   (6)
M_DONE                      =00000002   121   (2)   #-178   (2)
M_ERROR_FLAG                =00001000   143   (2)   #-554   (6)    #-558   (6)    #-563   (6)    #-567   (6)
                                                    #-581   (6)    #-583   (6)    #-585   (6)    #-587   (6)
                                                    #-589   (6)    #-591   (6)    #-593   (6)
M_EXECUTE_ERR               =00000004   123   (2)   #-1088  (11)   #-1208  (12)   #-1326  (13)   #-1445  (14)
                                                    #-1570  (16)   #-1656  (17)   #-1742  (18)   #-1831  (19)
```

```
                                                         #-1919    (20)    #-694     (8)     #-845     (9)     #-970     (10)
M_FATAL_ERR                     =00000100     135    (2)     #-580     (6)
M_GPR_ERR                       =00000400     139    (2)     #-554     (6)
M_ISP_ERR                       =00004000     147    (2)     #-567     (6)
M_ISTK                          =00010000     151    (2)     #-560     (6)
M_KSP_ERR                       =00002000     145    (2)     #-563     (6)
M_PSL_ERR                       =00000200     137    (2)     #-581     (6)
M_READY                         =00000001     119    (2)     #-1086    (11)    #-1206    (12)    #-1324    (13)    #-1443    (14)
                                                             #-1568    (16)    #-1654    (17)    #-1740    (18)    #-1829    (19)
                                                             #-1917    (20)    #-691     (8)     #-838     (9)     #-965     (10)
M_SP_ERR                        =00008000     149    (2)     #-558     (6)
M_TIMER                         =00000020     129    (2)     #-511     (6)
M_XFC                           =00000800     141    (2)     #-538     (6)
M_XFC_ERR                       =00000010     127    (2)     #-529     (6)
PCS                             =00000008     95     (2)
POWER_FAIL                      =00000003     95     (2)
PR$_ICCS                         00000000-XR                 #-510     (6)
PR$_ISP                          00000000-XR                 #-1095    (11)    #-1215    (12)    #-1333    (13)    #-1452    (14)
                                                             #-1495    (15)    #-1577    (16)    #-1663    (17)    #-1749    (18)
                                                             #-1839    (19)    #-500     (6)     #-617     (7)     #-854     (9)
                                                             #-979     (10)
PR$_KSP                          00000000-XR                 #-499     (6)
PR$_SCBB                         00000000-XR                 #-1826    (19)    #-1908    (20)
PRT$C_KR                         00000000-XR                 #-827     (9)
PRT$C_KW                         00000000-XR                 #-800     (9)     #-861     (9)
PSL$K_EXEC                      =00000001     92     (2)
PSL$K_KERNEL                    =00000000     92     (2)
PSL$K_SUPER                     =00000002     92     (2)
PSL$K_USER                      =00000003     92     (2)
PSL$M_CBIT                      =00000001     92     (2)
PSL$M_NBIT                      =00000008     92     (2)
PSL$M_VBIT                      =00000002     92     (2)
PSL$M_ZBIT                      =00000004     92     (2)
PSL$S_CURMOD                    =00000002                    #-570     (6)     #-572     (6)
PSL$V_CBIT                      =00000000     92     (2)
PSL$V_CURMOD                    =00000018                    #-570     (6)     #-572     (6)
PSL$V_IS                        =0000001A                    #-1069    (11)    #-1090    (11)    #-1189    (12)    #-1210    (12)
                                                             #-1307    (13)    #-1328    (13)    #-1426    (14)    #-1447    (14)
                                                             #-1572    (16)    #-1658    (17)    #-1744    (18)    #-1834    (19)
                                                             #-559     (6)     #-575     (6)     #-577     (6)     #-809     (9)
                                                             #-849     (9)     #-944     (10)    #-974     (10)
PSL$V_NBIT                      =00000003     92     (2)
PSL$V_VBIT                      =00000001     92     (2)
PSL$V_ZBIT                      =00000002     92     (2)
READ_GPR                         00000C6A-R   487    (6)       179     (2)
REG_TABLE                        00000000-XR                   417     (5)
RENTRY1                          00000159-R   695    (8)       676     (8)
RENTRY10                         000008EC-R   1305   (13)      1299    (13)
RENTRY11                         000009AD-R   1327   (13)      1312    (13)
RENTRY12                         00000A6C-R   1424   (14)      1418    (14)
RENTRY13                         00000B2D-R   1446   (14)      1431    (14)
RENTRY15                         0000013C-R   1571   (16)      1559    (16)
RENTRY16                         0000026E-R   1657   (17)      1645    (17)
RENTRY17                         000003A0-R   1743   (18)      1731    (18)
RENTRY18                         000004EF-R   1832   (19)      1817    (19)
```

```
RENTRY19                          0000062A-R   1920    (20)     1907    (20)
RENTRY2                           00000220-R   807     (9)      804     (9)
RENTRY3                           00000363-R   847     (9)      816     (9)
RENTRY4                           00000450-R   942     (10)     939     (10)
RENTRY5                           00000530-R   972     (10)     952     (10)
RENTRY6                           000005EC-R   1067    (11)     1061    (11)
RENTRY7                           000006AD-R   1089    (11)     1074    (11)
RENTRY8                           0000076C-R   1187    (12)     1181    (12)
RENTRY9                           0000082D-R   1209    (12)     1194    (12)
RENTRY_POINT                      00000004-R   99      (2)    #-1074    (11)    #-1194    (12)    #-1312    (13)    #-1431    (14)
                                                              #-1559    (16)    #-1645    (17)    #-1731    (18)    #-1817    (19)
                                                              #-1907    (20)      502     (6)     #-676     (8)     #-816     (9)
                                                              #-952     (10)
RESTART                          =00000100     177     (2)    #-1072    (11)    #-1073    (11)    #-1192    (12)    #-1193    (12)
                                                              #-1310    (13)    #-1311    (13)    #-1429    (14)    #-1430    (14)
                                                              #-1557    (16)    #-1558    (16)    #-1643    (17)    #-1644    (17)
                                                              #-1729    (18)    #-1730    (18)    #-1815    (19)    #-1816    (19)
                                                              #-1905    (20)    #-1906    (20)      448     (6)       449     (6)
                                                              #-674     (8)     #-675     (8)     #-814     (9)     #-815     (9)
                                                              #-950     (10)    #-951     (10)
SCB$L_CHME                        00000044                    #-1300    (13)    #-1340    (13)    #-1727    (18)    #-1755    (18)
SCB$L_CHMK                        00000040                    #-1419    (14)    #-1459    (14)    #-1813    (19)    #-1827    (19)
                                                              #-1845    (19)
SCB$L_CHMS                        00000048                    #-1182    (12)    #-1222    (12)    #-1641    (17)    #-1669    (17)
SCB$L_CHMU                        0000004C                    #-1062    (11)    #-1102    (11)    #-1555    (16)    #-1583    (16)
SCB$L_OPCCUS                      00000014                    #-1061    (11)    #-1101    (11)    #-1181    (12)    #-1221    (12)
                                                              #-1299    (13)    #-1339    (13)    #-1418    (14)    #-1458    (14)
                                                              #-1904    (20)    #-1927    (20)    #-804     (9)     #-812     (9)
                                                              #-813     (9)     #-866     (9)     #-939     (10)    #-947     (10)
                                                              #-949     (10)    #-987     (10)
SCB$L TIMER                       000000C0                    #-1461    (14)    #-1497    (15)    #-1929    (20)    #-619     (7)
SEP_FUNCTIONAL                   =00000002     73      (2)
SEP_REPAIR                       =00000003     73      (2)
SETUP RPB                         00000C00-R   445     (6)      1076    (11)      1196    (12)      1314    (13)      1433    (14)
                                                                678     (8)       819     (9)       955     (10)
SIZ...                           =00000001     92      (2)      73      (2)       92      (2)
SYS$SETPRT                        00010430                      800     (9)       827     (9)       861     (9)
T4_S1                             00000088-R   670     (8)
T4_S1_X                           000001A0-R   704     (8)
T4_S2                             000001AB-R   794     (9)
T4_S2_X                           0000040E-R   867     (9)
T4_S3                             00000419-R   934     (10)
T4_S3_X                           0000059A-R   988     (10)
T4_S4                             000005A5-R   1057    (11)
T4_S4_X                           0000071D-R   1103    (11)
T4_S5                             00000728-R   1177    (12)
T4_S5_X                           0000089D-R   1223    (12)
T4_S6                             000008A8-R   1295    (13)
T4_S6_X                           00000A1D-R   1341    (13)
T4_S7                             00000A28-R   1414    (14)
T4_S7_X                           00000B9D-R   1460    (14)
T5_S1                             00000080-R   1552    (16)
T5_S1_X                           000001A7-R   1584    (16)
T5_S2                             000001B2-R   1638    (17)
T5_S2_X                           000002D9-R   1670    (17)
```

```
T5_S3                           000002E4-R   1724   (18)
T5_S3_X                         0000040B-R   1756   (18)
T5_S4                           00000416-R   1810   (19)
T5_S4_X                         00000561-R   1846   (19)
T5_S5                           0000056C-R   1901   (20)
T5_S5_X                         00000674-R   1928   (20)
TEST_004                        00000030-R   611    (7)      611    (7)
TEST_004_X                      00000BB8-R   1462   (14)
TEST_005                        00000028-R   1486   (15)     1486   (15)
TEST_005_X                      0000068F-R   1931   (20)
TIMER_SERVICE                   00000CB8-R   509    (6)      1497   (15)     619    (7)
TU58                           =00000002     95     (2)
T_ACT_PSL                       000008E7-R   330    (4)      405    (5)
T_BAD_DATA                      000007D4-R   309    (4)
T_CHMESCB_ERR                   0000052B-R   264    (4)    #-1752   (18)
T_CHME_ERR                      000003DD-R   245    (4)    #-1336   (13)
T_CHMKSCB_ERR                   00000566-R   267    (4)    #-1842   (19)
T_CHMK_ERR                      00000405-R   248    (4)    #-1455   (14)
T_CHMSSCB_ERR                   000004F0-R   261    (4)    #-1666   (17)
T_CHMS_ERR                      000003B5-R   242    (4)    #-1218   (12)
T_CHMUSCB_ERR                   000004B5-R   258    (4)    #-1580   (16)
T_CHMU_ERR                      0000038D-R   239    (4)    #-1098   (11)
T_CHMX_ERR                      0000065B-R   282    (4)      379    (5)
T_CONTINUE                      00000212-R   216    (3)      1079   (11)     1199   (12)     1317   (13)     1436   (14)
                                                             1561   (16)     1647   (17)     1733   (18)     1819   (19)
                                                             1911   (20)     681    (8)      822    (9)      958    (10)
T_DBL_MCHK_ERR                  000000E6-R   195    (3)
T_EXECUTE                       00000601-R   276    (4)      377    (5)
T_EXITING                       000001C7-R   207    (3)
T_EXP_PSL                       000008C2-R   327    (4)      401    (5)
T_GOOD_DATA                     000007FB-R   312    (4)
T_GPR                           00000929-R   336    (4)      412    (5)
T_GPR_ERR                       000007AB-R   306    (4)
T_HALT_ERR                      000002F1-R   230    (4)    #-699    (8)
T_HALT_MESS                     000002C1-R   227    (4)      370    (5)
T_INSTRUCTIONS                  000001DC-R   211    (3)      1078   (11)     1198   (12)     1316   (13)     1435   (14)
                                                             1560   (16)     1646   (17)     1732   (18)     1818   (19)
                                                             1910   (20)     680    (8)      821    (9)      957    (10)
T_INVLDIS_ERR                   00000315-R   233    (4)    #-857    (9)
T_ISP_ERR1                      00000715-R   294    (4)      389    (5)
T_ISP_ERR2                      00000743-R   297    (4)      391    (5)
T_KSP_ERR1                      000006B9-R   288    (4)      384    (5)
T_KSP_ERR2                      000006E7-R   291    (4)      386    (5)
T_NOT_DONE                      000005D3-R   273    (4)      375    (5)
T_NOT_ON_IS                     0000089B-R   324    (4)      1070   (11)     1190   (12)     1308   (13)     1427   (14)
                                                             810    (9)      945    (10)
T_NOT_ON_KS                     00000877-R   321    (4)
T_NOT_READY                     000005A1-R   270    (4)      373    (5)
T_NOXOR                         0000096B-R   342    (4)      422    (5)
T_NO_OPER                       0000004D-R   190    (3)      1488   (15)     613    (7)
T_NXMIS_ERR                     00000350-R   236    (4)    #-983    (10)
T_NXM_WCS_ERR                   00000124-R   198    (3)
T_PAGEOPROT                     00000816-R   315    (4)      802    (9)
T_PAGEPROT                      00000844-R   318    (4)      829    (9)      863    (9)
T_PSL                           0000090C-R   333    (4)      410    (5)
```

```
T_REST_OFF                      00000197-R   204    (3)
T_SCBBNXM_ERR                   0000015B-R   201    (3)
T_SCB_ERR                       0000042D-R   251    (4)    #-1924   (20)
T_SP_ERR1                       00000771-R   300    (4)      393    (5)
T_SP_ERR2                       0000079F-R   303    (4)      395    (5)
T_TIMER                         00000691-R   285    (4)      425    (5)
T_WARNING                       00000262-R   220    (3)      1491   (15)      615     (7)
T_WCS_ERR                       00000463-R   254    (4)
T_XFC                           00000626-R   279    (4)      381    (5)
T_XOR                           00000951-R   339    (4)      420    (5)
V_BAD_MODE                     =00000006     130    (2)
V_BAD_STK                      =00000007     132    (2)
V_CHMX_ERR                     =00000003     124    (2)    #-378    (5)    #-588    (6)
V_DONE                         =00000001     120    (2)    #-374    (5)    #-586    (6)
V_ERROR_FLAG                   =0000000C     142    (2)    #-1097   (11)   #-1217   (12)   #-1335   (13)   #-1454   (14)
                                                           #-1579   (16)   #-1665   (17)   #-1751   (18)   #-1841   (19)
                                                           #-1923   (20)   #-698    (8)    #-856    (9)    #-982    (10)
V_EXECUTE_ERR                  =00000002     122    (2)    #-376    (5)    #-584    (6)
V_FATAL_ERR                    =00000008     134    (2)    #-1099   (11)   #-1219   (12)   #-1337   (13)   #-1456   (14)
                                                           #-1581   (16)   #-1667   (17)   #-1753   (18)   #-1843   (19)
                                                           #-1925   (20)   #-702    (8)    #-859    (9)    #-984    (10)
V_GPR_ERR                      =0000000A     138    (2)    #-411    (5)
V_ISP_ERR                      =0000000E     146    (2)    #-388    (5)
V_ISTK                         =00000010     150    (2)    #-382    (5)    #-387    (5)    #-564    (6)
V_KSP_ERR                      =0000000D     144    (2)    #-383    (5)
V_PSL_ERR                      =00000009     136    (2)    #-396    (5)
V_READY                        =00000000     118    (2)    #-372    (5)    #-582    (6)
V_SP_ERR                       =0000000F     148    (2)    #-392    (5)
V_TIMER                        =00000005     128    (2)    #-424    (5)    #-592    (6)
V_USR_WCS                      =00000019     154    (2)
V_XFC                          =0000000B     140    (2)
V_XFC_ERR                      =00000004     126    (2)    #-380    (5)    #-590    (6)
WCS                            =00000007     95     (2)
XFC_ERROR                       00000CCC-R   528    (6)      1904   (20)
XFC_SERVICE                     00000CD8-R   537    (6)      813    (9)        949    (10)
```

```
                                                  +--------------------------+
                                                  ! Macros Cross Reference !
                                                  +--------------------------+

MACRO                SIZE           DEFINITION       REFERENCES...
-----                ----           ----------       -------------
$D1_ABPROGRAM        1              614     (7)      1071    (11)      1100    (11)      1191    (12)      1220    (12)      1309    (13)
                                                     1338    (13)      1428    (14)      1457    (14)      1489    (15)      1582    (16)
                                                     1668    (17)      1754    (18)      1844    (19)      1926    (20)      614     (7)
                                                     701     (8)       703     (8)       803     (9)       811     (9)       830     (9)
                                                     858     (9)       860     (9)       864     (9)       946     (10)      985     (10)
$D1_BSUBT            1              670     (8)      1057    (11)      1177    (12)      1295    (13)      1414    (14)      1552    (16)
                                                     1638    (17)      1724    (18)      1810    (19)      1901    (20)      670     (8)
                                                     794     (9)       934     (10)
$D1_BTEST            1              611     (7)      1486    (15)      611     (7)
$D1_ERR              1              699     (8)      1070    (11)      1098    (11)      1190    (12)      1218    (12)      1308    (13)
                                                     1336    (13)      1427    (14)      1455    (14)      1580    (16)      1666    (17)
                                                     1752    (18)      1842    (19)      1924    (20)      699     (8)       802     (9)
                                                     810     (9)       829     (9)       857     (9)       863     (9)       945     (10)
                                                     983     (10)
$D1_ERR_S            1              699     (8)      1070    (11)      1098    (11)      1190    (12)      1218    (12)      1308    (13)
                                                     1336    (13)      1427    (14)      1455    (14)      1580    (16)      1666    (17)
                                                     1752    (18)      1842    (19)      1924    (20)      699     (8)       802     (9)
                                                     810     (9)       829     (9)       857     (9)       863     (9)       945     (10)
                                                     983     (10)
$D1_ESUBT            1              704     (8)      1103    (11)      1223    (12)      1341    (13)      1460    (14)      1584    (16)
                                                     1670    (17)      1756    (18)      1846    (19)      1928    (20)      704     (8)
                                                     867     (9)       988     (10)
$D1_ETEST            1              1462    (14)     1462    (14)      1931    (20)
$D1_PRINT_S          1              370     (5)      1078    (11)      1079    (11)      1198    (12)      1199    (12)      1316    (13)
                                                     1317    (13)      1435    (14)      1436    (14)      1488    (15)      1491    (15)
                                                     1560    (16)      1561    (16)      1646    (17)      1647    (17)      1732    (18)
                                                     1733    (18)      1818    (19)      1819    (19)      1910    (20)      1911    (20)
                                                     370     (5)       371     (5)       373     (5)       375     (5)       377     (5)
                                                     379     (5)       381     (5)       384     (5)       386     (5)       389     (5)
                                                     391     (5)       393     (5)       395     (5)       401     (5)       405     (5)
                                                     410     (5)       412     (5)       420     (5)       422     (5)       425     (5)
                                                     613     (7)       615     (7)       680     (8)       681     (8)       821     (9)
                                                     822     (9)       957     (10)      958     (10)
$D1_SBTTL            2              598     (7)      1107    (12)      1226    (13)      1344    (14)      1467    (15)      1501    (16)
                                                     1587    (17)      1673    (18)      1759    (19)      1850    (20)      598     (7)
                                                     623     (8)       707     (9)       870     (10)      991     (11)
$D2 BDATA            1              611     (7)      1486    (15)      611     (7)
$D2_BTEST            1              611     (7)      1486    (15)      611     (7)
$D2_SBTTL            1              598     (7)      1467    (15)      598     (7)
$DEF                 1              92      (2)
$DEFINI              1              88      (2)      88      (2)       91      (2)       92      (2)       93      (2)
$DS_ABORT            1              614     (7)      1071    (11)      1100    (11)      1191    (12)      1220    (12)      1309    (13)
                                                     1338    (13)      1428    (14)      1457    (14)      1489    (15)      1582    (16)
                                                     1668    (17)      1754    (18)      1844    (19)      1926    (20)      614     (7)
                                                     701     (8)       703     (8)       803     (9)       811     (9)       830     (9)
                                                     858     (9)       860     (9)       864     (9)       946     (10)      985     (10)
$DS BCOMPLETE        1              700     (8)      700     (8)       801     (9)       828     (9)       862     (9)
$DS BGNMESSAGE       1              367     (5)      367     (5)
```

```
$DS_BGNMOD        1             73     (2)       73     (2)
$DS_BGNSUB        1             670    (8)       1057   (11)      1177   (12)      1295   (13)      1414   (14)      1552   (16)
                                                 1638   (17)      1724   (18)      1810   (19)      1901   (20)      670    (8)
                                                 794    (9)       934    (10)
$DS_BGNTEST       1             611    (7)       1486   (15)      611    (7)
$DS_BOPER         1             612    (7)       1487   (15)      612    (7)
$DS_BREAK         1             1462   (14)      1462   (14)      1931   (20)
$DS_CLRVEC_S      1             812    (9)       1101   (11)      1102   (11)      1221   (12)      1222   (12)      1339   (13)
                                                 1340   (13)      1458   (14)      1459   (14)      1461   (14)      1583   (16)
                                                 1669   (17)      1755   (18)      1845   (19)      1927   (20)      1929   (20)
                                                 812    (9)       866    (9)       947    (10)      987    (10)
$DS_CVTREG_S      1             398    (5)       398    (5)       402    (5)       407    (5)
$DS_DSADEF        5             93     (2)       93     (2)
$DS_DSDEF         2             89     (2)       89     (2)
$DS_DSSDEF        8             91     (2)       91     (2)
$DS_ENDDATA       1             611    (7)       1486   (15)      611    (7)
$DS_ENDMESSAGE    1             427    (5)       427    (5)
$DS_ENDMOD        1             1933   (20)      1933   (20)
$DS_ENDSUB        1             704    (8)       1103   (11)      1223   (12)      1341   (13)      1460   (14)      1584   (16)
                                                 1670   (17)      1756   (18)      1846   (19)      1928   (20)      704    (8)
                                                 867    (9)       988    (10)
$DS_ENDTEST       1             1462   (14)      1462   (14)      1931   (20)
$DS_ENVDEF        2             73     (2)       73     (2)
$DS_ERRDEF        1             90     (2)       90     (2)
$DS_ERRHARD_S     1             699    (8)       1070   (11)      1098   (11)      1190   (12)      1218   (12)      1308   (13)
                                                 1336   (13)      1427   (14)      1455   (14)      1580   (16)      1666   (17)
                                                 1752   (18)      1842   (19)      1924   (20)      699    (8)       810    (9)
                                                 857    (9)       945    (10)      983    (10)
$DS_ERRSYS_S      1             802    (9)       802    (9)       829    (9)       863    (9)
$DS_MMOFF_S       1             865    (9)       865    (9)       986    (10)
$DS_MMON_S        1             799    (9)       799    (9)       938    (10)
$DS_PAGE          1             596    (6)       1104   (11)      1224   (12)      1342   (13)      1463   (14)      1499   (15)
                                                 1585   (16)      1671   (17)      1757   (18)      1847   (19)      1930   (20)
                                                 596    (6)       621    (7)       705    (8)       868    (9)       989    (10)
$DS_PRINTB_S      1             370    (5)       370    (5)       371    (5)       373    (5)       375    (5)       377    (5)
                                                 379    (5)       381    (5)       384    (5)       389    (5)       393    (5)
                                                 401    (5)       405    (5)       412    (5)       420    (5)       425    (5)
$DS_PRINTF_S      1             613    (7)       1078   (11)      1079   (11)      1198   (12)      1199   (12)      1316   (13)
                                                 1317   (13)      1435   (14)      1436   (14)      1488   (15)      1491   (15)
                                                 1560   (16)      1561   (16)      1646   (17)      1647   (17)      1732   (18)
                                                 1733   (18)      1818   (19)      1819   (19)      1910   (20)      1911   (20)
                                                 613    (7)       615    (7)       680    (8)       681    (8)       821    (9)
                                                 822    (9)       957    (10)      958    (10)
$DS_PRINTX_S      1             386    (5)       386    (5)       391    (5)       395    (5)       410    (5)       422    (5)
$DS_PSLDEF        1             92     (2)       92     (2)
$DS_SBTTL         1             598    (7)       1107   (12)      1226   (13)      1344   (14)      1467   (15)      1501   (16)
                                                 1587   (17)      1673   (18)      1759   (19)      1850   (20)      598    (7)
                                                 623    (8)       707    (9)       870    (10)      991    (11)
$DS_SCBDEF        3             88     (2)       88     (2)
$DS_SECDEF        1             95     (2)       95     (2)
$DS_SETVEC_S      1             619    (7)       1061   (11)      1062   (11)      1181   (12)      1182   (12)      1299   (13)
                                                 1300   (13)      1418   (14)      1419   (14)      1497   (15)      1555   (16)
                                                 1641   (17)      1727   (18)      1813   (19)      1904   (20)      619    (7)
                                                 804    (9)       813    (9)       939    (10)      949    (10)
$EQU              1             92     (2)       73     (2)       89     (2)       92     (2)
```

```
$EQULS1               1           92      (2)         73      (2)         89      (2)         92      (2)
$EQULST               1           73      (2)         73      (2)         89      (2)         92      (2)
$GBLINI               2           73      (2)         73      (2)         89      (2)         92      (2)
$OFFDEF               1           90      (2)         90      (2)
$PSLDEF               2           92      (2)         92      (2)
$PUSHADR              1           400     (5)         1070    (11)        1098    (11)        1190    (12)        1218    (12)        1308    (13)
                                                      1336    (13)        1427    (14)        1455    (14)        1580    (16)        1666    (17)
                                                      1752    (18)        1842    (19)        1924    (20)        400     (5)         404     (5)
                                                      409     (5)         699     (8)         800     (9)         802     (9)         810     (9)
                                                      827     (9)         829     (9)         857     (9)         861     (9)         863     (9)
                                                      945     (10)        983     (10)
$PUSHTWO              1           800     (9)         800     (9)         827     (9)         861     (9)
$SETPRT_S             1           800     (9)         800     (9)         827     (9)         861     (9)
$VIELD                1           73      (2)         73      (2)         92      (2)
$VIELD1               1           92      (2)         73      (2)         92      (2)
ECKAX_SECDEF          1           95      (2)         95      (2)

                                                +-------------------------+
                                                ! Performance indicators !
                                                +-------------------------+

Phase                        Page faults    CPU Time       Elapsed Time
-----                        -----------    --------       ------------
Initialization                       117    00:00:00.26    00:00:01.25
Command processing                   880    00:00:00.81    00:00:02.45
Pass 1                              1134    00:00:28.10    00:00:53.40
Symbol table sort                      0    00:00:00.86    00:00:00.93
Pass 2                                42    00:00:08.30    00:00:14.65
Symbol table output                    2    00:00:00.40    00:00:00.63
Psect synopsis output                  2    00:00:00.04    00:00:00.04
Cross-reference output                 7    00:00:03.16    00:00:05.48
Assembler run totals                2185    00:00:41.95    00:01:18.84

The working set limit was 3062 pages.
323343 bytes (632 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 574 non-local and 151 local symbols.
1934 source lines were read in Pass 1, producing 48 object records in Pass 2.
111 pages of virtual memory were used to define 55 macros.

                                               +--------------------------+
                                               ! Macro library statistics !
                                               +--------------------------+

Macro library name                               Macros defined
------------------                               --------------
DISK$DIAGPACK02:[TAYLOR.ECKAX]ECKAx.MLB;1                 1
SYS$COMMON:[SYSLIB]DIAG.MLB;955                          45
SYS$COMMON:[SYSLIB]STARLET.MLB;1                         10
TOTALS (all libraries)                                   56

828 GETS were required to define 56 macros.

There were no errors, warnings or information messages.

MACRO/LIS=ECKAX4.LIS/CROSS ECKAX4
```

Table of contents

```
    (1)        1        ECKAX5: WRITABLE CONTROL STORE
    (2)       63        DECLARATIONS
    (3)      220        SUBROUTINES
    (5)      448        ERROR MESSAGES
    (7)      517        ERROR REPORTING ROUTINES
    (8)      585        TEST 6:  WCS Memory Test
    (9)      620        SUBTEST 6.1:  Data Path
    (10)     698        SUBTEST 6.2:  Row Select
    (11)     766        SUBTEST 6.3:  Address Path
    (12)     831        SUBTEST 6.4:  Moving Inversions
    (13)     962        TEST 7:  WCS Execution Test
    (14)     990        SUBTEST 7.1:  WCS Entry/Exit
    (15)    1042        SUBTEST 7.2:  WCS 'Next Field' Addressing
```

```
0000     1          .SBTTL  ECKAX5: WRITABLE CONTROL STORE
0000     2          .TITLE  ECKAX5  VAX-11/750 Specific CPU Cluster Exerciser
0000     3          .IDENT  /5.2/
0000     4
0000     5 ;
0000     6 ; Copyright (C) 1980,1983,1986
0000     7 ; Digital Equipment Corporation, Maynard, Massachusetts 01754
0000     8 ;
0000     9 ; This software is furnished  under a license for use only on a single
0000    10 ; computer  system and  may be  copied only with  the inclusion of the
0000    11 ; above copyright notice.  This software, or any other copies thereof,
0000    12 ; may not be provided or  otherwise made available to any other person
0000    13 ; except for use on such system and to one who agrees to these license
0000    14 ; terms.  Title to and  ownership of the  software  shall at all times
0000    15 ; remain in DEC.
0000    16 ;
0000    17 ; The information in this software is subject to change without notice
0000    18 ; and should  not be construed  as a commitment  by Digital  Equipment
0000    19 ; Corporation.
0000    20 ;
0000    21 ; DEC assumes  no  responsibility  for  the use or  reliability of its
0000    22 ; software on equipment which is not supplied by DEC.
0000    23 ;
0000    24
0000    25 ;++
0000    26 ; Facility:     VAX Architecture Diagnostic.
0000    27 ;
0000    28 ; Abstract:     This module tests the writable contole store for parity
0000    29 ;               errors, addressing errors and sequence timing errors.
0000    30 ;
0000    31 ;
0000    32 ; Environment:  VAX Diagnostic Supervisor.
0000    33 ;
0000    34 ; Author:       Fred Roemer     10-Nov-80       Version 2.1
0000    35 ;
0000    36 ;       NO CHANGES TO THIS MODULE FOR VERSION 3.3 (DOCUMENT CHANGE ONLY)
0000    37 ;
0000    38 ;
0000    39 ;       Dan Milleville  19-Mar-82               Version 3.4
0000    40 ;       Changed Test 3 exit code to accrmodate temporarily a hardware problem.
0000    41 ;
0000    42 ;       Dave Shull      30-Nov-82               Version 4.0
0000    43 ;       Added tests 13 and 14 for coverage of PCS module.
0000    44 ;
0000    45 ;       Tai-Chun Pan    04-Mar-83               Version 5.0
0000    46 ;       Added test 5 for coverage of Processor Halts Architecture Undefined.
0000    47 ;       Added tests 11 and 12 for coverage of new ROM with MRSP of TU58.
0000    48 ;
0000    49 ;       Dan Milleville  21-Nov-83               Version 5.1
0000    50 ;       Fixed error in timer routine (didn't turn off timer).
0000    51 ;
0000    52 ;       Wayne R. Taylor February 1986           Version 5.2
0000    53 ;       Redefined IPR's ti jive with starlet.
0000    54 ;       Modified the code for problem report Eckax5 where:
0000    55 ;               Tests 6 & 13 tests with cache off. They want it on.
```

```
     0000     56 ;               Subtest 1 does not check bit 0 of the memory under test.
     0000     57 ;               Subtest 4 tests row 0, 1, 2, but not row 3.
     0000     58 ;       Modifications were supplied by the submitter.
     0000     59 ;       Search on 5.2
     0000     60 ;
     0000     61 ;--
```

```
                     0000    63          .SBTTL  DECLARATIONS
                     0000    64          .LIST   MEB
                     0000    65          .NLIST  CND
                 00000000    66          .PSECT  ECKAXE, PAGE, NOWRT
                     0000    67          .LIBRARY        'SYS$LIBRARY:DIAG'; VAX family diagnostic library.
                     0000    68          $DS_BGNMOD      CEP_REPAIR, TN=6
                     0000       ;;;      Macro-32 Diagnostic macro library Version 8.3 'DIAG.MLB (245)
                     0000    69
                     0000    70 ;
                     0000    71 ; Include files:
                     0000    72 ;
                     0000    73
                     0000    74          .LIBRARY        'ECKAX'         ; CPU Cluster Exerciser library.
                     0000    75
                     0000    79
                     0000    80          $DS_SCBDEF
                     0000    81          $DS_DSADEF
                     0000    82          $DS_DSDEF
                     0000    83          $DS_PARDEF
                     0000    84          $DS_PSLDEF
                     0000    85          $DS_ERRDEF
                     0000    86          $DS_DSSDEF
                     0000    87          $DS_HPODEF
                     0000    88          $DS_KA_DEF
                     0000    89          ECKAX_SECDEF
                     0000    90          $PR750DEF       ; 5.2
                     0000    91
                     0000    92 ;
                     0000    93 ; Own storage:
                     0000    94 ;
                     0000    95
         00000004    0000    96 L_COUNT:        .BLKL                   ; used as a counter
         00000008    0004    97 L_INC:          .BLKL                   ; used to store increment
         0000000C    0008    98 L_ROW:          .BLKL                   ; used to store current row
         00000010    000C    99 A_RETURN:       .BLKL                   ; used to store return address
                     0010   100
         00000014    0010   101 L_FLAGS:        .BLKL                   ; contains the following flags
                     0014   102
         00000000    0014   103 V_PAT=0                                 ; pattern done flag
         00000001    0014   104 M_PAT=1@0                               ;
         00000001    0014   105 V_DATA_ERR=1                            ; data error flag
         00000002    0014   106 M_DATA_ERR=1@1                          ;
         00000002    0014   107 V_ROW_ERR=2                             ; row error flag
         00000004    0014   108 M_ROW_ERR=1@2                           ;
         00000003    0014   109 V_REVERSE=3                             ; reverse direction flag
         00000008    0014   110 M_REVERSE=1@3                           ;
         00000004    0014   111 V_XFC_ERR=4                             ; unexpected XFC exception flag
         00000010    0014   112 M_XFC_ERR=1@4                           ;
                     0014   113
         00000000    0014   114 L_PERM_FLAGS:   .LONG   0               ; contains the following flags
                     0018   115
         00000000    0018   116 V_OLD_WCS=0                             ; old WCS module (ie., no PCS)
         00000001    0018   117 M_OLD_WCS=1@0
                     0018   118
         0000000E    0018   119 V_BIT14=14                              ; bit pos'tion for bit 14
```

```
0000000F  0018   120 V_BIT15=15
00080000  0018   121 M_BIT19=1@19                            ; bit mask for bit 19
00100000  0018   122 M_BIT20=1@20                            ; bit mask for bit 20
          0018   123
00F00000  0018   124 A_WCS_ADDRESS:  .LONG    ^XF00000       ; used to store address of WCS memory
          001C   125
00F0C000  001C   126 A_WCS_ENA_ADDR: .LONG    ^XF0C000       ; address of wcs write enable register
          0020   127
          0020   128 V_OVERFLOW_BIT:
0000000E  0020   129                 .LONG    14             ; used to store bit number for overflow
          0024   130                                         ; check
          0024   131
0000000A  0024   132 ADDR_SIZE:      .LONG    10             ; size of address vield for 1k WCS
          0028   133
          0028   134 WCS_MAX_SIZE_MWORDS:
000003FF  0028   135                 .LONG    1023           ; used to store WCS size
          002C   136
          002C   137 WCS_MAX_SIZE_LWORDS:
00000FFF  002C   138                 .LONG    4095           ; size of WCS in Longwords
          0030   139
          0030   140 MICRO_WORD_COUNT:
000003E5  0030   141                 .LONG    997            ; used to store number of microwords to
          0034   142                                         ; be loaded depending of size of WCS
          0034   143                                         ; (1k or 2k)
          0034   144 TEST_VALUE:
000003F5  0034   145                 .LONG    1013           ; used to store value that will be
          0038   146                                         ; use to decrement while running from
          0038   147                                         ; WCS
          0038   148
          0038   149
          0038   150 A_ENA_REGISTER:
00F0C000  0038   151                 .LONG    ^XF0C000       ; Register address to enable writing of
          003C   152                                         ; WCS
          003C   153
          003C   154 DATA_PATTERN:
00200000  003C   155                 .LONG    ^X200000       ; floating one pattern
FFDFFFFF  0040   156                 .LONG    ^XFFDFFFFF     ; floating zero pattern
          0044   157
000000C4  0044   158 A_WCS_BUF:      .BLKL    32             ; area used to store data
          00C4   159
00000144  00C4   160 DECODED_WORDS:  .BLKL    32             ; area used to store decoded data words
          0144   161
000001C4  0144   162 NOISY_WORDS:    .BLKL    32             ; area used to store encoded data words
          01C4   163
0C00FFFF  01C4   164 CODEVECTORS:    .LONG    ^XFFFF         ; identity vector
00005555  01C8   165                 .LONG    ^X5555         ; vector 1
00003333  01CC   166                 .LONG    ^X3333         ; vector 2
00000F0F  01D0   167                 .LONG    ^X0F0F         ; vector 3
          01D4   168
00000000  01D4   169 CODE_WORDS:     .LONG    ^X0000         ; encoded 0
0000FFFF  01D8   170                 .LONG    ^XFFFF         ; encoded 1
00005555  01DC   171                 .LONG    ^X5555         ; encoded 2
0000AAAA  01E0   172                 .LONG    ^XAAAA         ; encoded 3
00003333  01E4   173                 .LONG    ^X3333         ; encoded 4
0000CCCC  01E8   174                 .LONG    ^XCCCC         ; encoded 5
```

```
00006666  01EC   175                .LONG    ^X6666         ; encoded 6
00009999  01F0   176                .LONG    ^X9999         ; encoded 7
00000F0F  01F4   177                .LONG    ^X0F0F         ; encoded 8
0000F0F0  01F8   178                .LONG    ^XF0F0         ; encoded 9
00005A5A  01FC   179                .LONG    ^X5A5A         ; encoded 10(decimal)
          0200   180
00000000  0200   181 DATA_TABLE:    .LONG    ^X00000
00011111  0204   182                .LONG    ^X11111
00033333  0208   183                .LONG    ^X33333
00077777  020C   184                .LONG    ^X77777
000FFFFF  0210   185                .LONG    ^XFFFFF
000EEEEE  0214   186                .LONG    ^XEEEEE
000CCCCC  0218   187                .LONG    ^XCCCCC
00088888  021C   188                .LONG    ^X88888
00000000  0220   189                .LONG    ^X00000
          0224   190
          0224   191
          0224   192 MICRO_DATA:
          0224   193                                        ; entry and exit micro words
          0224   194                                        ; next field inverted in the first 20
0000BFC2  0224   195                .LONG    ^X0BFC2        ; bits. exit micro word at first loc.
00000487  0228   196                .LONG    ^X00487
00092403  022C   197                .LONG    ^X92403
000C81A5  0230   198                .LONG    ^XC81A5
00001FC2  0234   199                .LONG    ^X01FC2        ; entry micro word, at second location
00054047  0238   200                .LONG    ^X54047
000BE603  023C   201                .LONG    ^XBE603
000C2805  0240   202                .LONG    ^XC2805
          0244   203
          0244   204                                        ; these used to build rest of u words
          0244   205                                        ; next field not yet inverted and last
          0244   206                                        ; section sets parity depending on next
          0244   207                                        ; field odd or even.
00054047  0244   208                .LONG    ^X54047        ; common for second section of u word
00071203  0248   209                .LONG    ^X71203        ; common for third section of u word
00002003  024C   210                .LONG    ^X02003        ; next field for filling WCS
0000840E  0250   211                .LONG    ^X0840E        ; fourth section of u word
00002000  0254   212                .LONG    ^X02000        ; next field for going to exit point
          0258   213
          0258   214
          0258   215
```

```
                                   0258    220 .SBTTL  SUBROUTINES
                                   0258    221
                                   0258    222
                                   0258    223 ; This subroutine is used to clear the data buffer areas in the beginning
                                   0258    224 ; of this program.
                                   0258    225
                                   0258    226
                                   0258    227 CLEAR_BUFFER:
                 55    FDE8 CF  DE 0258    228         MOVAL   A_WCS_BUF,R5      ; get starting buffer address
                       FD9F CF  D4 025D    229         CLRL    L_COUNT           ; initialize counter
                             85 D4 0261    230 1$:     CLRL    (R5)+             ; clear a buffer location
      F4 FD94 CF    00000060 8F F3 0263    231         AOBLEQ  #96,L_COUNT,1$ ; br = not finished
                                05 026D    232         RSB                       ; return
                                   026E    233
                                   026E    234 ; This routine is used to initailly zero WCS
                                   026E    235
                                   026E    236 CLEAR_WCS:
            04 FDA1 CF    00    E0 026E    237         BBS     #V_OLD_WCS,L_PERM_FLAGS,10$; br = old WCS module
                       FDA4 DF  D4 0274    238         CLRL    @A_WCS_ENA_ADDR ; Enable writing of WCS
                 55    FD9C CF  D0 0278    239 10$:    MOVL    A_WCS_ADDRESS,R5     ; get starting WCS address
                       FD7F CF  D4 027D    240         CLRL    L_COUNT           ; initialize counter
                             65 D4 0281    241 20$:    CLRL    (R5)              ; clear a buffer location
                       55    04 C0 0283    242         ADDL2   #4,R5             ; get next address
         F3 FD73 CF    FDA2 CF  F3 0286    243         AOBLEQ  WCS_MAX_SIZE_LWORDS,L_COUNT,20$
                                   028E    244                                   ; br = not finished
                                05 028E    245         RSB                       ; return
                                   028F    246
                                   028F    247
                                   028F    248 ; This routine is used to decode data read from the WCS that has been encoded
                                   028F    249 ; in the REED - MULLER code for the address path subtest.
                                   028F    250
                                   028F    251 RM_DECODE:
                       03FF 8F  BB 028F    252         PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9>
                             52 D4 0293    253         CLRL    R2                ; initialize word counter
                       54    0B D0 0295    254         MOVL    #11,R4            ; initialize number of words to decode
                     FE27 CF42  D4 0298    255 10$:    CLRL    DECODED_WORDS[R2] ; initialize decoded word storage
                       53    01 D0 029D    256         MOVL    #1,R3             ; initialize the pair seperation
                       55    01 D0 02A0    257         MOVL    #1,R5             ; initialize the message bit counter
                       56    0F D0 02A3    258 20$:    MOVL    #15,R6            ; initialize the code word bit position
                             58 D4 02A6    259         CLRL    R8                ; initialize the population counter
                       55    01 D1 02A8    260         CMPL    #1,R5             ; if decoding bit one, skip next line
                             04 13 02AB    261         BEQL    30$               ; br = do not update pair seperation
                 53    53    01 78 02AD    262         ASHL    #1,R3,R3          ; update pair seperation
              57    FE8E CF42   D0 02B1    263 30$:    MOVL    NOISY_WORDS[R2],R7 ; get the encoded word
                    05 57    56 E1 02B7    264         BBC     R6,R7,40$         ; br = codeword bit not set and store zero
                       50    01 D0 02BB    265         MOVL    #1,R0             ; store a one for bit set in first of pair
                             02 11 02BE    266         BRB     50$               ; skip
                             50 D4 02C0    267 40$:    CLRL    R0                ; store a zero for bit clear
                       56    53 C2 02C2    268 50$:    SUBL2   R3,R6             ; point to the next bit of the pair
                    05 57    56 E1 02C5    269         BBC     R6,R7,60$         ; br = codeword bit not set and store zero
                       51    01 D0 02C9    270         MOVL    #1,R1             ; store a one for bit set in second of pair
                             02 11 02CC    271         BRB     70$               ; skip
                             51 D4 02CE    272 60$:    CLRL    R1                ; store a zero for bit clear
                       51    50 CC 02D0    273 70$:    XORL2   R0,R1             ; compute the value of the relation
                             02 13 02D3    274         BEQL    80$               ; br = value is not a one
```

```
                      58   D6  02D5           INCL    R8                  ; bump the population counter
           50    01   55   78  02D7   80$:    ASHL    R5,#1,R0            ; calculate 2**R5 for message bit
      59   00    56   01   7A  02DB           EMUL    #1,R6,#0,R9         ;
      50   50    59   50   7B  02E0           EDIV    R0,R9,R0,R0         ;
                      50   D5  02E5           TSTL    R0                  ;
                      03   13  02E7           BEQL    90$                 ; br =
                 56   53   C0  02E9           ADDL2   R3,R6               ;
                   C2 56   F4  02EC   90$:    SOBGEQ  R6,30$              ;
                 04   58   D1  02EF           CMPL    R8,#4               ;
                      09   15  02F2           BLEQ    100$                ;
             FDCB CF42     DF  02F4           PUSHAL  DECODED_WORDS[R2] ;
              00 9E   55   E2  02F9           BBSS    R5,@(SP)+,100$    ;
              A2 55   03   F3  02FD   100$:   AOBLEQ  #3,R5,20$           ; br = not finished with message bits
              93 52   54   F2  0301           AOBLSS  R4,R2,10$           ; br = not finished with all codewords
                      52   D4  0305           CLRL    R2                  ; intialize word counter to do bit 0
                 54   0B   D0  0307           MOVL    #11,R4              ; initialize number of words to decode
         50  FE35 CF42     D0  030A   110$:   MOVL    NOISY_WORDS[R2],R0 ; get encoded word
                 53   01   D0  0310           MOVL    #1,R3               ; initialize bit counter
         56  FDAC CF42     D0  0313   120$:   MOVL    DECODED_WORDS[R2],R6 ; get a partially decoded word
      56   56    01   53   EF  0319           EXTZV   R3,#1,R6,R6         ; get a decoded bit
         56  FEA1 CF43     C4  031E           MULL2   CODEVECTORS[R3],R6 ; get product of vector and decoded bit
                 50   56   CC  0324           XORL2   R6,R0               ; get XOR of each product
              E8 53   03   F3  0327           AOBLEQ  #3,R3,120$          ; do for all three decoded bits
                      55   D4  032B           CLRL    R5                  ; initialize the population counter
                      53   D4  032D           CLRL    R3                  ; initialize the bit counter
              02 50   53   E1  032F   130$:   BBC     R3,R0,140$          ; br = bit is a zero
                      55   D6  0333           INCL    R5                  ; else bump the population counter
              F6 53   0F   F3  0335   140$:   AOBLEQ  #15,R3,130$         ; do for all 16 bits
                 08   55   D1  0339           CMPL    R5,#8               ; make the majority decision
                      09   15  033C           BLEQ    150$                ; br = majority not ones
             FD81 CF42     DF  033E           PUSHAL  DECODED_WORDS[R2] ; prepare to set LSB
              00 9E   00   E2  0343           BBSS    #0,@(SP)+,150$    ; set LSB in decoded word
              BF 52   54   F2  0347   150$:   AOBLSS  R4,R2,110$          ; br = not finished with all words
                03FF  8F   BA  034B           POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9>
                           05  034F           RSB                         ; return
                               0350
                               0350
                               0350   ; This routine is used to load a 1K buffer area with test addresses for the
                               0350   ; moving inversions subtest. The different sequence of addresses accessed
                               0350   ; for the current increment are computed and written into the buffer area.
                               0350
                               0350   SETUP_ADDR:
           57    FCB0 CF   D0  0350           MOVL    L_INC,R7            ; save increment
      58    00000000'EF    D0  0355           MOVL    A_WCS_BUFFER,R8 ; save address of buffer to store addresses
           5B    FCB8 CF   D0  035C           MOVL    A_WCS_ADDRESS,R11 ; save first address of wcs
      59   5B    FCA3 CF   C1  0361   5$:     ADDL3   L_ROW,R11,R9        ; add row being tested to address to R9
                 88   59   D0  0367   10$:    MOVL    R9,(R8)+            ; load address to use into buffer
                 59   57   C0  036A           ADDL2   R7,R9               ; add increment to get next address
        F4 59    FCAF CF   E1  036D           BBC     V_OVERFLOW_BIT,R9,10$
                               0373                                       ; br = no address overflow yet
                 59   57   C2  0373           SUBL2   R7,R9               ; get last address done
  5A    59  FCA9 CF   04   EE  0376           EXTV    #4,ADDR_SIZE,R9,R10
                               037D                                       ; get address by itself
      5A    FFFFFFFF  8F   D1  037D           CMPL    #-1,R10             ; check for minus one
                      05   13  0384           BEQL    20$                 ; br = finished building addresses
```

```
                  5B    10    C0  0386    330         ADDL2   #16,R11             ; increment the base address
                        D6    11  0389    331         BRB     5$                  ; go finish building addresses
                              05  038B    332 20$:    RSB                         ; return
                                  038C    333
                                  038C    334 ; This subroutine will load WCS with executable micro code. The value in
                                  038C    335 ; L_COUNT minus 3 will determine the number of micro words that will be loaded.
                                  038C    336 ; For the entry/exit test a count of zero is used.
                                  038C    337
                                  038C    338 LOAD_MICRO:
          04 FC83 CF    00    E0  038C    339         BBS     #V_OLD_WCS,L_PERM_FLAGS,1$; br = old WCS module
                  FC86 DF     D4  0392    340         CLRL    @A_WCS_ENA_ADDR ; Enable writing of WCS
                        5A    D4  0396    341 1$:     CLRL    R10                 ; initialize a counter for first two words
             53   FC7C CF     D0  0398    342         MOVL    A_WCS_ADDRESS,R3 ; load WCS starting address
             54   FE83 CF     DE  039D    343         MOVAL   MICRO_DATA,R4       ; load address of micro code to be loaded
                  63    84    D0  03A2    344 2$:     MOVL    (R4)+,(R3)          ; load 20 bits of micro word
                  53    04    C0  03A5    345         ADDL2   #4,R3               ; get next address
               F6 5A    08    F2  03A8    346         AOBLSS  #8,R10,2$           ; br = until first and last words are done
                        56    D4  03AC    347         CLRL    R6                  ; initialize next address field for other words
                  57    84    7D  03AE    348         MOVQ    (R4)+,R7            ; load constant second 40 bits
                  56    84    D0  03B1    349         MOVL    (R4)+,R6            ; setup next address field
                  FC48 CF     D5  03B4    350         TSTL    L_COUNT             ; check for extra words
                        44    13  03B8    351         BEQL    30$                 ; br = only one other word to load
                        50    7C  03BA    352 3$:     CLRQ    R0                  ; init bit and count to figure high parity bit
                  59    64    D0  03BC    353 4$:     MOVL    (R4),R9             ; load last 20 bits, parity bit not set
               02 56    50    E1  03BF    354         BBC     R0,R6,10$           ; br = bit is zero do not count
                        51    D6  03C3    355         INCL    R1                  ; bit non zero, count
               F3 50    0C    F2  03C5    356 10$:    AOBLSS  #12,R0,4$           ; br = not finished checking next address field
                     07 51    E8  03C9    357         BLBS    R1,20$              ; br = odd number of bits, do not set parity
       59    00080000 8F      C8  03CC    358         BISL2   #M_BIT19,R9         ; set high parity bit
                  52    56    D0  03D3    359 20$:    MOVL    R6,R2               ; load next address field into R2
      5B    52    08    06    EF  03D6    360         EXTZV   #6,#8,R2,R11        ; get part of next field to invert
                  5B    5B    B2  03DB    361         MCOMW   R11,R11             ; invert the data
      52    08    06    5B    F0  03DE    362         INSV    R11,#6,#8,R2        ; put complete next address field together
                  63    52    D0  03E3    363         MOVL    R2,(R3)             ; write first 20 bits
               04 A3    57    D0  03E6    364         MOVL    R7,4(R3)            ; write second 20 bits
               08 A3    58    D0  03EA    365         MOVL    R8,8(R3)            ; write third 20 bits
               0C A3    59    D0  03EE    366         MOVL    R9,12(R3)           ; write last 20 bits
                  53    10    C0  03F2    367         ADDL2   #16,R3              ; update address for next word
                        56    D6  03F5    368         INCL    R6                  ; setup next address field
               BE FC05 CF     F5  03F7    369         SOBGTR  L_COUNT,3$          ; br = load next micro word
                        07    19  03FC    370         BLSS    40$                 ; br = finished
               56    04 A4    D0  03FE    371 30$:    MOVL    4(R4),R6            ; setup last next address field
                     FFB5     31  0402    372         BRW     3$                  ; go load it
   FC0A DF   00100000 8F      C8  0405    373 40$:    BISL2   #M_BIT20,@A_WCS_ADDRESS ; enable WCS
                              05  040E    374         RSB                         ; return
                                  040F    375
                                  040F    376 ; This routine is used to service unxepected XFC exceptions.
                                  040F    377
                                  040F    378 .ALIGN  LONG
                                  0410    379
                                  0410    380 XFC_SERVICE:
             FBFB CF    10    C8  0410    381         BISL2   #M_XFC_ERR,L_FLAGS ; set error flag
                              02  0415    382         REI                         ; return
                                  0416    383
                                  0416    384 ; This routine is used to service expected and unexpected machine check
```

```
                                     0416   385 ; exceptions. Unexpected machine check exceptions are resignaled and
                                     0416   386 ; reported by the Supervisor.
                                     0416   387
                                     0416   388 .ALIGN  LONG
                                     0418   389
                                     0418   390 CHECK_MCHK:
                    04 AE   01   D1  0418   391         CMPL    #1,4(SP)        ; check for CS parity error
                            16   13  041C   392         BEQL    1$              ; br = yes parity error
                            0E   DD  041E   393         PUSHL   #14             ; put size on stack for printsig
                   00660088'EF   DD  0420   394         PUSHL   DS$_MCHK        ; and type
                                     0426   395         $DS_PRINTSIG_G ARGPTR=(SP)
              00010100 9F   6E   FA  0426                       CALLG   (SP), a#DS$PRINTSIG
                                     042D   396         $DS_ABORT
              00010020 9F   6C   FA  042D                       CALLG   (AP), a#DS$ABORT
                    5E      8E   C0  0434   397 1$:     ADDL2   (SP)+,SP        ; get over machine check logout
                  6E    FBD1 CF  D0  0437   398         MOVL    A_RETURN,(SP)   ; set up return address
                                 02  043C   399         REI                     ; return
                                     043D   400
                                     043D   401 ;
                                     043D   402 ; This subroutine will check to see what size of WCS is installed on the system
                                     043D   403 ; (ie., 1k or 2k) and then set-up the appropriate data structures.  If the
                                     043D   404 ; WCS is equal to 1k then just return as the default value is for 1k.  The
                                     043D   405 ; thing that is check is to see if the microcode revision level is greater
                                     043D   406 ; than 94(decimal) and if not then this has to be a 1k module.  If it is
                                     043D   407 ; greater then 94 then this could be a 2k WCS.
                                     043D   408 ;
                                     043D   409
                                     043D   410 SIZE_WCS:
              55   00000000'8F   DB  043D   411         MFPR    #PR$_SID,R5                     ; get system id register
             56   55   08   08   EF  0444   412         EXTZV   #8,#8,R5,R6                     ; extract the micro code revision level
              00000000'EF   56   D1  0449   413         CMPL    R6,L_REV_CHECK                  ; check the revision level
                            05   1A  0450   414         BGTRU   5$                              ; br = this is a L0008 module
                  FBBD CF   01   C8  0452   415         BISL2   #M_OLD_WCS,L_PERM_FLAGS ; set flag specifying old WCS (L0005)
              51   00000000'EF   D0  0457   416 5$:     MOVL    KA_PTABLE,R1                    ; get address of the P table
          52   36 A1   0C   00   EF  045E   417         EXTZV   #0,#12,KA$W_WCS(R1),R2  ; Get WCS last address, only last
                                     0464   418                                         ; three digits (ie., 2'3FF')
              52   000003FF 8F   D1  0464   419         CMPL    #1023,R2                        ; check to see if 1k module
                            67   13  046B   420         BEQLU   20$                             ; Branch to return if 1k module as the
                                     046D   421                                         ; data structures default to 1k module
               1C FBA2 CF   00   E0  046D   422         BBS     #V_OLD_WCS,L_PERM_FLAGS,9$; br = old WCS module, if the program
                                     0473   423                                         ; falls through to here then the attach
                                     0473   424                                         ; was wrong (ie., the old wcs is 1k and
                                     0473   425                                         ; the attach specified something other
                                     0473   426                                         ; then 1k - 23FF).
              52   000007FF 8F   D1  0473   427         CMPL    #2047,R2                        ; check to see if 2k module
                            26   13  047A   428         BEQLU   10$                             ; go set-up data structures for 2k
                                     047C   429                                         ; module
                                     047C   430 7$:     $DS_PRINTF_S    FORMAT=T_BAD_ATTACH
                   00000568'EF   9F  047C                       PUSHAB  T_BAD_ATTACH
              000100F0 9F   01   FB  0482                       CALLS   #$$N, a#DS$PRINTF
                                     0489   431         $DS_EXIT        TEST
                       50   01   D0  0489                       MOVL    #1,R0           ; NOT AN ERROR
                          FF38'  31  048C                       BRW     TEST_006_X      ; EXIT TEST 6
                                     048F   432 9$:     $DS_PRINTF_S    FORMAT=T_BAD_ATTACH2
                   000005DB'EF   9F  048F                       PUSHAB  T_BAD_ATTACH2
              000100F0 9F   01    FB  0495                       CALLS   #$$N, @#DS$PRINTF
                                      049C   433         $DS_EXIT        TEST
                       50   01    D0  049C                       MOVL    #1,R0           ; NOT AN ERROR
                          FF25'   31  049F                       BRW     TEST_006_X      ; EXIT TEST 6
                  FB79 CF   0F    D0  04A2   434 10$:    MOVL    #V_BIT15,V_OVERFLOW_BIT ; Set overflow check to bit 15
         FB78 CF   000007FF 8F    D0  04A7   435         MOVL    #2047,WCS_MAX_SIZE_MWORDS
                                      04B0   436                                         ; Max size = 2047 in Microwords
         FB73 CF   00001FFF 8F    D0  04B0   437         MOVL    #8191,WCS_MAX_SIZE_LWORDS
                                      04B9   438                                         ; Max size = 8191 in Longwords
         FB6E CF   000007E5 8F    D0  04B9   439         MOVL    #2021,MICRO_WORD_COUNT  ; micro word count = 2021
         FB69 CF   000007F5 8F    D0  04C2   440         MOVL    #2037,TEST_VALUE        ; test value = 2037
                  FB54 CF   0B    D0  04CB   441         MOVL    #11,ADDR_SIZE           ; Address size = 11
                       FB64 DF    D4  04D0   442         CLRL    @A_ENA_REGISTER         ; Enable writing of WCS
                                  05  04D4   443 20$:    RSB                             ; return
                                      04D5   444
                                      04D5   445
```

```
                                     04D5    447
                                     04D5    448 .SBTTL  ERROR MESSAGES
                                     04D5    449
                                     04D5    450 ;-----------------------------------------------------------------------------
                                     04D5    451 T_CS_PARITY1:
65 20 79 74 69 72 61 70 20 53 43 00' 04D5    452          .ASCIC   CS parity error occured durring moving inversions.
64 65 72 75 63 63 6F 20 72 6F 72 72  04E1
76 6F 6D 20 67 6E 69 72 72 75 64 20  04ED
6F 69 73 72 65 76 6E 69 20 67 6E 69  04F9
                            2E 73 6E 0505
                                  32 04D5
                                     0508    453 ;-----------------------------------------------------------------------------
                                     0508    454 T_CS_PARITY2:
65 20 79 74 69 72 61 70 20 53 43 00' 0508    455          .ASCIC   CS parity error occured durring entry/exit.
64 65 72 75 63 63 6F 20 72 6F 72 72  0514
74 6E 65 20 67 6E 69 72 72 75 64 20  0520
            2E 74 69 78 65 2F 79 72  052C
                                  2B 0508
                                     0534    456 ;-----------------------------------------------------------------------------
                                     0534    457 T_CS_PARITY3:
65 20 79 74 69 72 61 70 20 53 43 00' 0534    458          .ASCIC   CS parity error occured durring next address field.
64 65 72 75 63 63 6F 20 72 6F 72 72  0540
78 65 6E 20 67 6E 69 72 72 75 64 20  054C
69 66 20 73 73 65 72 64 64 61 20 74  0558
                         2E 64 6C 65 0564
                                  33 0534
                                     0568    459 ;-----------------------------------------------------------------------------
                                     0568    460 T_BAD_ATTACH:
65 72 64 64 61 20 53 43 57 2F 21 00' 0568    461          .ASCIC   !/WCS address Not Equal to 23FF, for 1k WCS, or 27FF, !/ -
6C 61 75 71 45 20 74 6F 4E 20 73 73  0574
6F 66 20 2C 46 46 33 32 20 6F 74 20  0580
72 6F 20 2C 53 43 57 20 6B 31 20 72  058C
72 6F 66 2F 21 20 2C 46 46 37 32 20  0598
75 64 6F 6D 20 53 43 57 20 6B 32 20  05A4
61 74 74 41 2D 65 52 20 20 2E 65 6C  05B0
72 72 6F 63 20 68 74 69 77 20 68 63  05BC
64 64 61 20 74 73 61 6C 20 74 63 65  05C8
               2F 21 2E 73 73 65 72  05D4
                                  72 0568
                                     05DB    462                   for 2k WCS module.  Re-Attach with correct last address.!/
                                     05DB    463 ;-----------------------------------------------------------------------------
                                     05DB    464 T_BAD_ATTACH2:
20 74 73 61 6C 20 53 43 57 2F 21 00' 05DB    465          .ASCIC   !/WCS last address Not Equal to 23FF for 1k WCS.!/ -
20 74 6F 4E 20 73 73 65 72 64 64 61  05E7
46 33 32 20 6F 74 20 6C 61 75 71 45  05F3
53 43 57 20 6B 31 20 72 6F 66 20 46  05FF
                            2F 21 2E 060B
69 77 20 68 63 61 74 74 41 2D 65 52  060E    466                   Re-Attach with correct last address.!/
6C 20 74 63 65 72 72 6F 63 20 68 74  061A
2E 73 73 65 72 64 64 61 20 74 73 61  0626
                               2F 21 0632
                                  58 05DB
                                     0634    467 ;------------------------------------------------------------------------  --  -
                                     0634    468 T_NO_RESPONCE:
61 20 74 73 72 69 66 20 53 43 57 00' 0634    469          .ASCIC   WCS first address did not respond when accessed.
```

```
6E 20 64 69 64 20 73 73 65 72 64 64  0640
77 20 64 6E 6F 70 73 65 72 20 74 6F  064C
64 65 73 73 65 63 63 61 20 6E 65 68  0658
                                 2E  0664
                                 30  0634
                                     0665   470 ;-------------------------------------------------------------------------
                                     0665   471 T_NO_WCS:
57 20 67 6E 69 74 69 78 45 2F 21 00' 0665   472         .ASCIC  !/Exiting WCS test, WCS last address = 0.!/
53 43 57 20 2C 74 73 65 74 20 53 43  0671
73 65 72 64 64 61 20 74 73 61 6C 20  067D
            2F 21 2E 30 20 3D 20 73  0689
                                 2B  0665
                                     0691   473 ;-------------------------------------------------------------------------
                                     0691   474 T_DATAPATH:
65 72 20 61 74 61 64 20 64 61 42 00' 0691   475         .ASCIC  Bad data recieved checking data path.
6B 63 65 68 63 20 64 65 76 65 69 63  069D
74 61 70 20 61 74 61 64 20 67 6E 69  06A9
                              2E 68  06B5
                                 25  0691
                                     06B7   476 ;-------------------------------------------------------------------------
                                     06B7   477 T_DPERR:
21 20 3D 20 73 73 65 72 64 64 41 00' 06B7   478         .ASCIC  Address = !XL  Expected = !XL  Actual = !XL  XOR = !XL!/
64 65 74 63 65 70 78 45 20 20 4C 58  06C3
75 74 63 41 20 20 4C 58 21 20 3D 20  06CF
4F 58 20 20 4C 58 21 20 3D 20 6C 61  06DB
         2F 21 4C 58 21 20 3D 20 52  06E7
                                 38  06B7
                                     06F0   479 ;-------------------------------------------------------------------------
                                     06F0   480 T_ROW:
65 72 20 61 74 61 64 20 64 61 42 00' 06F0   481         .ASCIC  Bad data recieved checking row select.
6B 63 65 68 63 20 64 65 76 65 69 63  06FC
65 6C 65 73 20 77 6F 72 20 67 6E 69  0708
                           2E 74 63  0714
                                 26  06F0
                                     0717   482 ;-------------------------------------------------------------------------
                                     0717   483 T_ROWERR:
20 20 20 73 73 65 72 64 64 41 20 00' 0717   484         .ASCIC   Address     Expected     Actual     XOR!/
20 20 64 65 74 63 65 70 78 45 20 20  0723
20 20 20 6C 61 75 74 63 41 20 20 20  072F
               2F 21 52 4F 58 20 20  073B
                                 2A  0717
                                     0742   485 ;-------------------------------------------------------------------------
                                     0742   486 F_ROWERR1:
4C 58 21 20 20 20 20 4C 58 21 20 00' 0742   487         .ASCIC   !XL    !XL    !XL   !XL!/
58 21 20 20 20 4C 58 21 20 20 20 20  074E
                           2F 21 4C  075A
                                 1A  0742
                                     075D   488 ;-------------------------------------------------------------------------
                                     075D   489 F_ROWERR2:
4C 58 21 20 20 20 20 4C 58 21 20 00' 075D   490         .ASCIC   !XL    !XL    !XL!/
         2F 21 4C 58 21 20 20 20 20  0769
                                 14  075D
                                     0772   491 ;-------------------------------------------------------------------------
                                     0772   492 T_ADR_ERR:
65 64 20 61 74 61 64 20 64 61 42 00' 0772   493         .ASCIC  Bad data decoded while checking address path.
```

```
20 65 6C 69 68 77 20 64 65 64 6F 63   077E
64 64 61 20 67 6E 69 6B 63 65 68 63   078A
      2E 68 74 61 70 20 73 73 65 72   0796
                                 2D   0772
                                      07A0     494 ;---------------------------------------------------
                                      07A0     495 T_MI_ERR1:
6E 69 20 61 74 61 64 20 64 61 42 00'  07A0     496          .ASCIC   Bad data in WCS befor writing for moving inversions.'
77 20 72 6F 66 65 62 20 53 43 57 20   07AC
6D 20 72 6F 66 20 67 6E 69 74 69 72   07B8
73 72 65 76 6E 69 20 67 6E 69 76 6F   07C4
                  2E 73 6E 6F 69      07D0
                                 34   07A0
                                      07D5     497 ;---------------------------------------------------
                                      07D5     498 T_MI_ERR2:
6E 69 20 61 74 61 64 20 64 61 42 00'  07D5     499          .ASCIC   Bad data in WCS after writing for moving inversions."
77 20 72 65 74 66 61 20 53 43 57 20   07E1
6D 20 72 6F 66 20 67 6E 69 74 69 72   07ED
73 72 65 76 6E 69 20 67 6E 69 76 6F   07F9
                  2E 73 6E 6F 69      0805
                                 34   07D5
                                      080A     500 ;---------------------------------------------------
                                      080A     501 T_ENTRY_ERR:
6E 69 20 61 74 61 64 20 64 61 42 00'  080A     502          .ASCIC   Bad data in R5 executing entry/exit.'
6E 69 74 75 63 65 78 65 20 35 52 20   0816
74 69 78 65 2F 79 72 74 6E 65 20 67   0822
                                 2E   082E
                                 24   080A
                                      082F     503 ;---------------------------------------------------
                                      082F     504 T_NXFIELD_ERR:
6E 69 20 61 74 61 64 20 64 61 42 00'  082F     505          .ASCIC   Bad data in R5 while checking WCS NEXT field.
68 63 20 65 6C 69 68 77 20 35 52 20   083B
4E 20 53 43 57 20 67 6E 69 6B 63 65   0847
      2E 64 6C 65 69 66 20 54 58 45   0853
                                 2D   082F
                                      085D     506 ;---------------------------------------------------
                                      085D     507 F_WCS_ERR:
45 20 2D 20 61 74 61 64 20 35 52 00'  085D     508          .ASCIC   R5 data - Expected = !XL  Actual = !XL  XOR = !XL!/'
58 21 20 3D 20 64 65 74 63 65 70 78   0869
20 3D 20 6C 61 75 74 63 41 20 20 4C   0875
21 20 3D 20 52 4F 58 20 20 4C 58 21   0881
                     2F 21 4C 58      088D
                                 33   085D
                                      0891     509 ;---------------------------------------------------
                                      0891     510 T_XFC_ERR:
69 20 79 72 74 6E 65 20 43 46 58 00'  0891     511          .ASCIC   'XFC entry in SCB went into memory instead of WCS.!/
69 20 74 6E 65 77 20 42 43 53 20 6E   089D
69 20 79 72 6F 6D 65 6D 20 6F 74 6E   08A9
43 57 20 66 6F 20 64 61 65 74 73 6E   08B5
                     2F 21 2E 53      08C1
                                 33   0891
                                      08C5     512 ;---------------------------------------------------
                                      08C5     513
                                      08C5     514
```

```
                                     08C5   516
                                     08C5   517 .SBTTL  ERROR REPORTING ROUTINES
                                     08C5   518
                                     08C5   519 ;-------------------------------------------------------------------------------
                                     08C5   520 DPERR:
                                     08C5   521         $DS_BGNMESSAGE  REGMASK=<R3,R4,R5,R6,R7>
                               00F8  08C5                   .WORD   ^M<R3,R4,R5,R6,R7>               ; ENTRY MASK
          57   18 AC   14   00   EF  08C7   522         EXTZV   #0,#20,ERR$_P2(AP),R7    ; get expected data
               53   1C AC   57   CD  08CD   523         XORL3   R7,ERR$_P3(AP),R3        ; get XOR of data
                                     08D2   524         $DS_PRINTB_S    FORMAT=T_DPERR,P1=ERR$_P1(AP),P2=R7,P3=ERR$_P3(AP),-
                                     08D2   525                         P4=R3
                            53   DD  08D2                   PUSHL   R3
                         1C AC   DD  08D4                   PUSHL   ERR$_P3(AP)
                            57   DD  08D7                   PUSHL   R7
                         14 AC   DD  08D9                   PUSHL   ERR$_P1(AP)
                       FDD7 CF   9F  08DC                   PUSHAB  T_DPERR
              000100E0 9F   05   FB  08E0                   CALLS   #$$N, @#DS$PRINTB
                                     08E7   526         $DS_ENDMESSAGE
                                 04  08E7                   RET                      ; RETURN TO DIAGNOSTIC SUPERVISOR
                                     08E8   527 ;-------------------------------------------------------------------------------
                                     08E8   528 ROWERR:
                                     08E8   529         $DS_BGNMESSAGE  REGMASK=<R4,R5,R8,R9>
                               0330  08E8                   .WORD   ^M<R4,R5,R8,R9>          ; ENTRY MASK
                                     08EA   530         $DS_PRINTB_S    FORMAT=T_ROWERR
                       FE29 CF   9F  08EA                   PUSHAB  T_ROWERR
              000100E0 9F   01   FB  08EE                   CALLS   #$$N, @#DS$PRINTB
                       58   01   D0  08F5   531         MOVL    #1,R8                    ; load expected data and counter
                  54   F71C CF   D0  08F8   532         MOVL    A_WCS_ADDRESS,R4         ; load initial WCS address
                  55   F743 CF   DE  08FD   533         MOVAL   A_WCS_BUF,R5             ; load address of buffer
                  59   65   58   CD  0902   534 1$:     XORL3   R8,(R5),R9               ; generate XOR of data
                            15   13  0906   535         BEQL    2$                       ; skip basic message if data is good
                                     0908   536         $DS_PRINTB_S    FORMAT=F_ROWERR1,P1=R4,P2=R8,P3=(R5),P4=R9
                            59   DD  0908                   PUSHL   R9
                            65   DD  090A                   PUSHL   (R5)
                            58   DD  090C                   PUSHL   R8
                            54   DD  090E                   PUSHL   R4
                       FE2E CF   9F  0910                   PUSHAB  F_ROWERR1
              000100E0 9F   05   FB  0914                   CALLS   #$$N, @#DS$PRINTB
                            11   11  091B   537         BRB     3$                       ; skip extended message
                                     091D   538 2$:     $DS_PRINTX_S    FORMAT=F_ROWERR2,P1=R4,P2=R8,P3=(R5)
                            65   DD  091D                   PUSHL   (R5)
                            58   DD  091F                   PUSHL   R8
                            54   DD  0921                   PUSHL   R4
                       FE36 CF   9F  0923                   PUSHAB  F_ROWERR2
              000100E8 9F   04   FB  0927                   CALLS   #$$N, @#DS$PRINTX
                            85   D5  092E   539 3$:     TSTL    (R5)+                    ; update R5
                       54   04   C0  0930   540         ADDL2   #4,R4                    ; update R4
                    CB 58   04   F3  0933   541         AOBLEQ  #4,R8,1$                 ; br = not finished
                                     0937   542         $DS_ENDMESSAGE
                                 04  0937                   RET                      ; RETURN TO DIAGNOSTIC SUPERVISOR
                                     0938   543 ;-------------------------------------------------------------------------------
                                     0938   544 ADDR_ERR:
                                     0938   545         $DS_BGNMESSAGE  REGMASK=<R4,R5,R6,R7,R8,R9>
                               03F0  0938                   .WORD   ^M<R4,R5,R6,R7,R8,R9>            ; ENTRY MASK
                                     093A   546         $DS_PRINTB_S    FORMAT=T_ROWERR
```

```
             FDD9 CF    9F  093A                          PUSHAB  T_ROWERR
      000100E0 9F   01    FB  093E                          CALLS   #$$N, @#DS$PRINTB
                    57    D4  0945   547         CLRL     R7                        ; load expected data and counter
        54    14 AC       D0  0947   548         MOVL     ERR$_P1(AP),R4            ; load first address of row selected
              53    08    D0  094B   549         MOVL     #8,R3                     ; initialize floating address
      55    F772 CF       DE  094E   550         MOVAL    DECODED_WORDS,R5          ; load address of actual data
      56    54    53      C1  0953   551 1$:     ADDL3    R3,R4,R6                  ; build address
      59    65    57      CD  0957   552         XORL3    R7,(R5),R9                ; generate XOR of data
                    15    13  095B   553         BEQL     2$                        ; skip basic message if data is good
                              095D   554         $DS_PRINTB_S     FORMAT=F_ROWERR1,P1=R6,P2=R7,P3=(R5),P4=R9
                    59    DD  095D                          PUSHL   R9
                    65    DD  095F                          PUSHL   (R5)
                    57    DD  0961                          PUSHL   R7
                    56    DD  0963                          PUSHL   R6
             FDD9 CF      9F  0965                          PUSHAB  F_ROWERR1
      000100E0 9F   05    FB  0969                          CALLS   #$$N, @#DS$PRINTB
                    11    11  0970   555         BRB      3$                        ; skip extended message
                              0972   556 2$:     $DS_PRINTX_S     FORMAT=F_ROWERR2,P1=R6,P2=R7,P3=(R5)
                    65    DD  0972                          PUSHL   (R5)
                    57    DD  0974                          PUSHL   R7
                    56    DD  0976                          PUSHL   R6
             FDE1 CF      9F  0978                          PUSHAB  F_ROWERR2
      000100E8 9F   04    FB  097C                          CALLS   #$$N, @#DS$PRINTX
                    85    D5  0983   557 3$:     TSTL     (R5)+                     ; update R5
      53    53    01      9C  0985   558         ROTL     #1,R3,R3                  ; float one across address
         C6 57    0A      F3  0989   559         AOBLEQ   #10,R7,1$                 ; br = not finished
                              098D   560         $DS_ENDMESSAGE
                          04  098D                          RET                     ; RETURN TO DIAGNOSTIC SUPERVISOR
                              098E   561 ;-------------------------------------------------------------------------------------
                              098E   562 MI_ERR:
                              098E   563         $DS_BGNMESSAGE   REGMASK=<R3,R4,R5,R6>
                        0078  098E                          .WORD   ^M<R3,R4,R5,R6>          ; ENTRY MASK
        54    14 AC       D0  0990   564         MOVL     ERR$_P1(AP),R4            ; get address of error
        55    18 AC       D0  0994   565         MOVL     ERR$_P2(AP),R5            ; get expected data
        53    1C AC       D0  0998   566         MOVL     ERR$_P3(AP),R3            ; get actual data
      56    53    55      CD  099C   567         XORL3    R5,R3,R6                  ; get XOR of data
                              09A0   568         $DS_PRINTB_S     FORMAT=T_DPERR,P1=R4,P2=R5,P3=R3,P4=R6
                    56    DD  09A0                          PUSHL   R6
                    53    DD  09A2                          PUSHL   R3
                    55    DD  09A4                          PUSHL   R5
                    54    DD  09A6                          PUSHL   R4
             FD0B CF      9F  09A8                          PUSHAB  T_DPERR
      000100E0 9F   05    FB  09AC                          CALLS   #$$N, @#DS$PRINTB
                              09B3   569         $DS_ENDMESSAGE
                          04  09B3                          RET                     ; RETURN TO DIAGNOSTIC SUPERVISOR
                              09B4   570 ;-------------------------------------------------------------------------------------
                              09B4   571 WCS_ERR:
                              09B4   572         $DS_BGNMESSAGE   REGMASK=<R3,R4,R5>
                        0038  09B4                          .WORD   ^M<R3,R4,R5>             ; ENTRY MASK
        55    14 AC       D0  09B6   573         MOVL     ERR$_P1(AP),R5            ; get actual data
              53    0F    D0  09BA   574         MOVL     #15,R3                    ; get expected data
      54    55    53      CD  09BD   575         XORL3    R3,R5,R4                  ; get XOR of data
                              09C1   576         $DS_PRINTB_S     FORMAT=F_WCS_ERR,P1=R3,P2=ERR$_P1(AP),P3=R4
                    54    DD  09C1                          PUSHL   R4
              14 AC       DD  09C3                          PUSHL   ERR$_P1(AP)
```

```
                       53   DD  09C6                        PUSHL   R3
                  FE91 CF   9F  09C8                        PUSHAB  F_WCS_ERR
         000100E0 9F    04  FB  09CC                        CALLS   #$$N, @#DS$PRINTB
          0B F638 CF    04  E1  09D3    577         BBC     #V_XFC_ERR,L_FLAGS,1$    ; br = no XFC error
                                09D9    578         $DS_PRINTB_S    FORMAT=T_XFC_ERR
                  FEB4 CF   9F  09D9                        PUSHAB  T_XFC_ERR
         000100E0 9F    01  FB  09DD                        CALLS   #$$N, @#DS$PRINTB
                                09E4    579 1$:
                                09E4    580         $DS_ENDMESSAGE
                            04  09E4                        RET                      ; RETURN TO DIAGNOSTIC SUPERVISOR
                                09E5    581 ;--------------------------------------------------------------------------------------------------
                                09E5    582
                                09E5    583         $DS_PAGE
```

```
                                     09E5               .SBTTL  TEST 6:  WCS Memory Test
                                 00000000                       .PSECT  TEST_006, PAGE, NOWRT
                                     0014
                                     0014   586 ;++
                                     0014   587 ; Test description:
                                     0014   588 ;
                                     0014   589 ;       This test checks the WCS as a parity memory with standard memory tests.
                                     0014   590 ;       These tests include floating ones and zeros on the data path, testing
                                     0014   591 ;       group select logic , floating ones and zeros on the address lines and
                                     0014   592 ;       testing the RAMs using the moving inversions algorithm.
                                     0014   593 ;
                                     0014   594 ;
                                     0014   595 ;--
                                     0014   596
                                     0014   597         $DS_BGNTEST     <WCS,DEFAULT,ALL>
                                     0014       DATA_006:
                           00000000  0014                       .LONG   0                ; TEST ARGUMENT TABLE TERMINATOR
                                     0018       TEST_006::
                               0000  0018                       .WORD   ^M<>             ; ENTRY MASK
              51   00000000'EF   D0  001A   598         MOVL    KA_PTABLE,R1             ; get address of P table
                         36 A1   B5  0021   599         TSTW    KA$W_WCS(R1)             ; check for WCS present
                            1C   12  0024   600         BNEQ    2$                       ; br = yes test WCS
              0000FE54'EF   01   D1  0026   601         CMPL    #1,DSA$GL_PASSNO         ; ckeck for first pass
                            0D   12  002D   602         BNEQ    1$                       ; br = not first pass
                                     002F   603         $DS_PRINTF_S    FORMAT=T_NO_WCS
                   00000665'EF   9F  002F                       PUSHAB  T_NO_WCS
              000100F0 9F   01   FB  0035                       CALLS   #$$N, @#DS$PRINTF
                                     003C   604 1$:     $DS_EXIT           TEST
                       50   01   D0  003C                       MOVL    #1,R0            ; NOT AN ERROR
                          0385   31  003F                       BRW     TEST_006_X       ; EXIT TEST 6
                                     0042   605 ;2$:    MTPR    #1,#PR$_CADR             ; Disable Cache for data test 5.2
                       25   00   DA  0042   606 2$:     MTPR    #0,#PR750$_CADR          ; Enable Cache for data test
              54   00000018'EF   D0  0045   607         MOVL    A_WCS_ADDRESS,R4
                                     004C   608         $DS_PROBE_S     ADDRESS=(R4),LENGTH=#4,UNIT=#0
                            00   DD  004C                       PUSHL   #0
                            04   DD  004E                       PUSHL   #4
                            64   9F  0050                       PUSHAB  (R4)
              000101A0 9F   03   FB  0052                       CALLS   #3, @#DS$PROBE
                         13 50   E8  0059   609         BLBS    R0,3$                    ; br = address responded
                                     005C   610         $DS_ERRHARD_S   UNIT=#0,MSGADR=T_NO_RESPONCE
                            00   DD  005C                       PUSHL   #0
                   00000634'EF   DF  005E                       PUSHAL  T_NO_RESPONCE
                            00   DD  0064                       PUSHL   #0
                            01   DD  0066                       PUSHL   #$ER
                                     0068       ;;; TEST 6, SUBTEST 0, ERROR 1
              000100D0 9F   04   FB  0068                       CALLS   #$$M, @#DS$ERRHARD
                                     006F   611 3$:     $DS_CKLOOP      2$
           00010040 9F   D0 AF   FA  006F                       CALLG   2$, @#DS$CKLOOP
                                     0077   612
                   00000014'EF   D4  0077   613         CLRL    L_PERM_FLAGS             ; Clear test - permanent flags at start
                   0000043D'EF   16  007D   614         JSB     SIZE_WCS                 ; Size up the WCS for either 1k or 2k
                   0000026E'EF   16  0083   615         JSB     CLEAR_WCS                ; Clear out WCS
     00000018'FF   00100000 8F   CA  0089   616         BICL2   #M_BIT20,@A_WCS_ADDRESS ; disable WCS
                                     0094   617
                                     0094   618         $DS_PAGE
```

```
                              0094            .SBTTL  SUBTEST 6.1:  Data Path
                              0094     621 ;+
                              0094     622 ; Subtest description:
                              0094     623 ;
                              0094     624 ;       The 11/750 micro word is 80 bits wide. When the WCS is addressed as a
                              0094     625 ;       memory, the 80 bit micro word is divided into four 1K by 20 bit rows
                              0094     626 ;       of words. This subtest checks the data path for stuck at one, stuck at
                              0094     627 ;       zero, and shorted data lines. This is accomplished by floating a one
                              0094     628 ;       and a zero through the data path for each 20 bit row of RAMs. If an
                              0094     629 ;       error is detected it is reported with information describing the nature
                              0094     630 ;       of the error. Any errors detected in this subtest must be corrected
                              0094     631 ;       befor going on to the next subtest.
                              0094     632 ;
                              0094     633 ;
                              0094     634 ; Subtest steps:
                              0094     635 ;
                              0094     636 ;       BGNSUBTEST Data Path
                              0094     637 ;       :  Initialize flags
                              0094     638 ;       :  Initialize data pattern
                              0094     639 ;       :  REPEAT
                              0094     640 ;       :  :  REPEAT
                              0094     641 ;       :  :  :  Write pattern
                              0094     642 ;       :  :  :  Read pattern
                              0094     643 ;       :  :  :  IF pattern incorrect
                              0094     644 ;       :  :  :  THEN
                              0094     645 ;       :  :  :  :  Report error
                              0094     646 ;       :  :  :  ENDIF
                              0094     647 ;       :  :  :  Update pattern
                              0094     648 ;       :  :  : UNTIL all patterns done
                              0094     649 ;       :  :  ENDREPEAT
                              0094     650 ;       :  : UNTIL all rows done
                              0094     651 ;       :  ENDREPEAT
                              0094     652 ;       ENDSUBTEST Data Path
                              0094     653 ;
                              0094     654 ;
                              0094     655 ; Errors:
                              0094     656 ;
                              0094     657 ;
                              0094     658 ;       If an error occurs, it can be helpful to allow the program to
                              0094     659 ;       continue running to collect data from other failures within this subtest
                              0094     660 ;       to determine the type of failure that is occuring. An example of this
                              0094     661 ;       would be to collect a set of error reports which indicate the same bit
                              0094     662 ;       error for all four rows. This might indicate that the error is in the
                              0094     663 ;       data path and not an individual RAM error.
                              0094     664 ;
                              0094     665 ;
                              0094     666 ;-
                              0094     667
                              0094     668
                              0094     669            $DS_BGNSUB
                              0094         T6_S1::
    00010030 9F    00000000'EF  FA  0094                   CALLG   $S$, a#DS$BGNSUB
                   00000010'EF  D4  009F     670       CLRL    L_FLAGS                 ; initialize flags
          54       00000018'EF  D0  00A5     671       MOVL    A_WCS_ADDRESS,R4        ; load first WCS address
                             58  D4  00AC     672       CLRL    R8                      ; initialize row counter
```

```
           55   0000003C'EF   D0  00AE    673 1$:      MOVL    DATA_PATTERN,R5          ; load all zeros pattern
                          57   D4  00B5    674 2$:      CLRL    R7                       ; initialize bit counter
            55    55    FF 8F   9C  00B7    675 3$:      ROTL    #-1,R5,R5                ; float pattern across data
          53    55    14    00   EF  00BC    676 4$:      EXTZV   #0,#20,R5,R3             ; extract the data to be written
                    64    53   D0  00C1    677          MOVL    R3,(R4)                  ; write data
                    56    64   D0  00C4    678          MOVL    (R4),R6                  ; read data
          56    56    14    00   EF  00C7    679          EXTZV   #0,#20,R6,R6             ; extract the data read
                    56    53   D1  00CC    680          CMPL    R3,R6                    ; compare data
                          25   13  00CF    681          BEQL    5$                       ; br = data good
                               00D1    682          $DS_ERRHARD_S   UNIT=#0,MSGADR=T_DATAPATH,PRLINK=DPERR,-
                               00D1    683                          P1=R4,P2=R3,P3=R6
                          56   DD  00D1                     PUSHL   R6
                          53   DD  00D3                     PUSHL   R3
                          54   DD  00D5                     PUSHL   R4
                000008C5'EF   DF  00D7                     PUSHAL  DPERR
                00000691'EF   DF  00DD                     PUSHAL  T_DATAPATH
                          00   DD  00E3                     PUSHL   #0
                          01   DD  00E5                     PUSHL   #$ER
                               00E7         ;;; TEST 6, SUBTEST 1, ERROR 1
        000100D0 9F       07   FB  00E7                     CALLS   #$$M, @#DS$ERRHARD
                               00EE    684          $DS_CKLOOP      4$               ; set loop for write and read
     00010040 9F      CB AF   FA  00EE                     CALLG   4$, @#DS$CKLOOP
                               00F6    685 ;5$:     AOBLEQ  #19,R7,3$                ; br = not done floating data 5.2
                 BD 57    14   F3  00F6    686 5$:      AOBLEQ  #20,R7,3$                ; br = not done floating data
     10 00000010'EF    00   E0  00FA    687          BBS     #V_PAT,L_FLAGS,6$        ; br = both data patterns done
        55    00000040'EF   D0  0102    688          MOVL    DATA_PATTERN+4,R5        ; point R5 to second pattern
        00000010'EF    01   88  0109    689          BISB2   #M_PAT,L_FLAGS           ; set data pattern done flag
                          A3   11  0110    690          BRB     2$                       ; go write second pattern
                    54    04   C0  0112    691 6$:      ADDL2   #4,R4                    ; select next row
        00000010'EF    01   CA  0115    692          BICL2   #M_PAT,L_FLAGS           ; clear data pattern done flag
                 8E 58    03   F3  011C    693          AOBLEQ  #3,R8,1$                 ; br = not done for all rows
                               0120    694          $DS_ENDSUB
                               0120         T6_S1_X:
 00010038 9F    00000000'EF   FA  0120                     CALLG   $S$, @#DS$ENDSUB
                               012B    695
                               012B    696          $DS_PAGE
```

```
                                             012B            .SBTTL  SUBTEST 6.2:  Row Select
                                             012B     699 ;+
                                             012B     700 ; Subtest description:
                                             012B     701 ;
                                             012B     702 ;        This subtest checks row select (bits 2 and 3 of the address). The WCS
                                             012B     703 ;        is addressed as four 1K by 20 bit rows of words. A unique pattern is
                                             012B     704 ;        written to each row, using the addresses that were tested and verrified
                                             012B     705 ;        in the data path subtest. Then each location is read back to see if the
                                             012B     706 ;        pattern is correct. If one of the row select lines are stuck one, stuck
                                             012B     707 ;        zero or shorted together, the data will be over written in at least one
                                             012B     708 ;        of the four rows. This subtest is dependant on the successful
                                             012B     709 ;        completion of the data path subtest.
                                             012B     710 ;
                                             012B     711 ;
                                             012B     712 ; Subtest steps:
                                             012B     713 ;
                                             012B     714 ;        BNGSUBTEST Row Select
                                             012B     715 ;        :  Write data patterns
                                             012B     716 ;        :  Read data patterns
                                             012B     717 ;        :  IF data patterns are incorrect
                                             012B     718 ;        :  :  THEN
                                             012B     719 ;        :  :  Report error
                                             012B     720 ;        :  ENDIF
                                             012B     721 ;        ENDSUBTEST Row Select
                                             012B     722 ;
                                             012B     723 ;
                                             012B     724 ; Errors:
                                             012B     725 ;
                                             012B     726 ;
                                             012B     727 ;        When an error is reported, the contents of the four addresses written
                                             012B     728 ;        and read will be displayed. An example of an error could be the low
                                             012B     729 ;        order bit of the row select is stuck zero. This means that instead of
                                             012B     730 ;        selecting row 0, 1, 2, and 3 (each written with a unique pattern), row
                                             012B     731 ;        0, 1, 0, 1 will be selected. When the data is read back row 0 will have
                                             012B     732 ;        pattern 3, row 1 will have pattern 4, and rows 3 and 4 will have
                                             012B     733 ;        correct data. After an error has been corrected in this subtest,
                                             012B     734 ;        testing must must continue with the data path subtest.
                                             012B     735 ;
                                             012B     736 ;-
                                             012B     737            $DS_BGNSUB
                                             012B         T6_S2::
      00010030 9F     0000000C'EF     FA     012B                     CALLG   $S$, a#DS$BGNSUB
                      00000010'EF     D4     0136     738            CLRL    L_FLAGS          ; initialize flags
                      00000258'EF     16     013C     739            JSB     CLEAR_BUFFER     ; zero buffer areas
                54    00000018'EF     D0     0142     740            MOVL    A_WCS_ADDRESS,R4 ; load first test address
                            58    01  D0     0149     741            MOVL    #1,R8            ; initialize row counter
                            64    58  D0     014C     742 1$:        MOVL    R8,(R4)          ; write data
                            54    04  C0     014F     743            ADDL2   #4,R4            ; update address for next row
                         F6 58    04  F3     0152     744            AOBLEQ  #4,R8,1$         ; br = not all rows done
                54    00000018'EF     D0     0156     745            MOVL    A_WCS_ADDRESS,R4 ; load first address to read
                            58    01  D0     015D     746            MOVL    #1,R8            ; initialize row counter
                55    00000044'EF     DE     0160     747            MOVAL   A_WCS_BUF,R5     ; load address of buffer for data
                            53    64  D0     0167     748 2$:        MOVL    (R4),R3          ; read data
                53     53   14    00  EF     016A     749            EXTZV   #0,#20,R3,R3     ; extract data from word written
                            85    53  D0     016F     750            MOVL    R3,(R5)+         ; put data into buffer
```

```
                       54   04   C0  0172   751         ADDL2  #4,R4           ; update address for next row
                    EE 58   04   F3  0175   752         AOBLEQ  #4,R8,2$        ; br = not all rows done
                       58   01   D0  0179   753         MOVL    #1,R8           ; initialize data and row counter
              55   00000044'EF   DE  017C   754         MOVAL   A_WCS_BUF,R5    ; load address of buffer
                       58   85   D1  0183   755 3$:     CMPL    (R5)+,R8        ; check data
                            07   13  0186   756         BEQL    4$              ; br = data error
              00000010'EF   04   C8  0188   757         BISL2   #M_ROW_ERR,L_FLAGS ; set error flag
                    F0 58   04   F3  018F   758 4$:     AOBLEQ  #4,R8,3$        ; br = not all rows done
           17 00000010'EF   02   E1  0193   759         BBC     #V_ROW_ERR,L_FLAGS,5$ ; br = no error
                                     019B   760         $DS_ERRHARD_S   UNIT=#0,MSGADR=T_ROW,PRLINK=ROWERR
                   000008E8'EF   DF  019B                       PUSHAL  ROWERR
                   000006F0'EF   DF  01A1                       PUSHAL  T_ROW
                            00   DD  01A7                       PUSHL   #0
                            01   DD  01A9                       PUSHL   #$ER
                                     01AB          ;;; TEST 6, SUBTEST 2, ERROR 1
              000100D0 9F   04   FB  01AB                       CALLS   #$$M, @#DS$ERRHARD
                                     01B2   761 5$:
                                     01B2   762         $DS_ENDSUB
                                     01B2          T6_S2_X:
     00010038 9F   0000000C'EF   FA  01B2                       CALLG   $S$, @#DS$ENDSUB
                                     01BD   763
                                     01BD   764         $DS_PAGE
                                     01BD                .SBTTL  SUBTEST 6.3:  Address Path
                                     01BD   767 ;+
                                     01BD   768 ; Subtest description:
                                     01BD   769 ;
                                     01BD   770 ;       This subtest checks the address path for bits stuck at one, stuck at
                                     01BD   771 ;       zero and shorted together. The WCS has a ten bit address path which
                                     01BD   772 ;       runs in parallel with all of the rams in all four rows. Each row is
                                     01BD   773 ;       tested seperatly by writing an encoded incremental pattern to a set of
                                     01BD   774 ;       addresses, reading the data back, decoding it and checking
                                     01BD   775 ;       for errors. The code used is the Reed - Muller (32,6) code which allows
                                     01BD   776 ;       for up to six RAM failures befor detecting an error. This method aides
                                     01BD   777 ;       in distingushing RAM errors from address path errors. Floating a one
                                     01BD   778 ;       across the address path makes up the set of addresses used in this
                                     01BD   779 ;       subtest.
                                     01BD   780 ;
                                     01BD   781 ;
                                     01BD   782 ; Subtest steps:
                                     01BD   783 ;
                                     01BD   784 ;       BGNSUBTEST Address Path
                                     01BD   785 ;       :  Initialize flags
                                     01BD   786 ;       :  REPEAT
                                     01BD   787 ;       :  :  Write encoded data to all addresses
                                     01BD   788 ;       :  :  Read encoded data from all addresses
                                     01BD   789 ;       :  :  Decode data
                                     01BD   790 ;       :  :  Check data
                                     01BD   791 ;       :  :  IF any data errors
                                     01BD   792 ;       :  :  :  THEN
                                     01BD   793 ;       :  :  :  Report data path error
                                     01BD   794 ;       :  :  ENDIF
                                     01BD   795 ;       :  : UNTIL all four rows done
                                     01BD   796 ;       :  ENDREPEAT
                                     01BD   797 ;       ENDSUBTEST Address Path
                                     01BD   798 ;
                                     01BD   799 ;
                                     01BD   800 ;-
                                     01BD   801
                                     01BD   802
                                     01BD   803         $DS_BGNSUB
                                     01BD       T6_S3::
     00010030 9F   00000018'EF   FA  01BD                       CALLG   $S$, @#DS$BGNSUB
                   00000010'EF   D4  01C8   804         CLRL    L_FLAGS                  ; init flags
                            54   D4  01CE   805         CLRL    R4                       ; initialize row counter
              58   00000018'EF   D0  01D0   806         MOVL    A_WCS_ADDRESS,R8         ; load first WCS address
                       55   58   D0  01D7   807 1$:     MOVL    R8,R5                    ; load it in R5 too
                       53   08   D0  01DA   808         MOVL    #8,R3                    ; initialize floating address
              56   000001D4'EF   DE  01DD   809         MOVAL   CODE_WORDS,R6            ; load address of encoded data
              57   00000144'EF   DE  01E4   810         MOVAL   NOISY_WORDS,R7           ; load address of data buffer
                       65   86   D0  01EB   811 2$:     MOVL    (R6)+,(R5)               ; write test data
                       87   65   D0  01EE   812         MOVL    (R5),(R7)+               ; read test data
                  53   53   01   9C  01F1   813         ROTL    #1,R3,R3                 ; float one across address
                  55   58   53   C1  01F5   814         ADDL3   R3,R8,R5                 ; build next test address
           EA 53   00000020'EF   E1  01F9   815         BBC     V_OVERFLOW_BIT,R3,2$     ; br = not finished
                   0000028F'EF   16  0201   816         JSB     RM_DECODE                ; go decode test data
                            59   D4  0207   817         CLRL    R9                       ; init counter
              5A   000000C4'EF   DE  0209   818         MOVAL   DECODED_WORDS,R10        ; load address of decoded words
```

```
                   8A   59   D1  0210    819 3$:      CMPL     R9,(R10)+                   ; check data
                        07   13  0213    820          BEQL     4$                          ; br = no error
            00000010'EF 02   C8  0215    821          BISL2    #M_DATA_ERR,L_FLAGS         ; set error flag
                  F0 59 0A   F3  021C    822 4$:      AOBLEQ   #10,R9,3$                   ; br = not finished checking data
         19 00000010'EF 01   E1  0220    823          BBC      #V_DATA_ERR,L_FLAGS,5$      ; br = no error
                                 0228    824          $DS_ERRHARD_S   UNIT=#0,MSGADR=T_ADR_ERR,PRLINK=ADDR_ERR,P1=R8
                        58   DD  0228                          PUSHL    R8
                00000938'EF  DF  022A                          PUSHAL   ADDR_ERR
                00000772'EF  DF  0230                          PUSHAL   T_ADR_ERR
                        00   DD  0236                          PUSHL    #0
                        01   DD  0238                          PUSHL    #$ER
                                 023A         ;;; TEST 6, SUBTEST 3, ERROR 1
            000100D0 9F 05   FB  023A                          CALLS    #$$M, @#DS$ERRHARD
                   58   04   C0  0241    825 5$:      ADDL2    #4,R8                       ; update row select
                  8F 54 03   F3  0244    826          AOBLEQ   #3,R4,1$                    ; br = not all rows done
                                 0248    827          $DS_ENDSUB
                                 0248         T6_S3_X:
     00010038 9F    00000018'EF  FA  0248                      CALLG    $S$, @#DS$ENDSUB
                                 0253    828
                                 0253    829          $DS_PAGE
```

```
0253             .SBTTL  SUBTEST 6.4:  Moving Inversions
0253     832 ;+
0253     833 ; Subtest description:
0253     834 ;
0253     835 ;       This subtest does an exhaustive test of all RAM chips using the
0253     836 ;       moving inversions algorithem. Each address accessed in this subtest
0253     837 ;       will be read, written and then read again with data which is checked
0253     838 ;       after each read. This subtest starts at the first address and works
0253     839 ;       its way up thru WCS memory, eventually writing all locations with all
0253     840 ;       ones. Then the process is started again and is complete when all
0253     841 ;       locations have been written with all zeros. Next this is done working
0253     842 ;       from the last address to the first, both writting till the data is all
0253     843 ;       ones and then all zeros. This is done for all bit positions in all RAMS
0253     844 ;       (four bits). For each successive pass a different address bit is used
0253     845 ;       as the lease signifigant bit in counting such that every address line
0253     846 ;       is toggled at the fastest possible rate.
0253     847 ;
0253     848 ;
0253     849 ; Subtest steps:
0253     850 ;
0253     851 ;       BGNSUBTEST Moving Inversions
0253     852 ;       : REPEAT
0253     853 ;       : : Set increment to 1
0253     854 ;       : : REPEAT
0253     855 ;       : : : Initialize flags
0253     856 ;       : : : Point to first entry in data table
0253     857 ;       : : : Set forward direction
0253     858 ;       : : : REPEAT
0253     859 ;       : : : : REPEAT
0253     860 ;       : : : : : Set address to starting address
0253     861 ;       : : : : : REPEAT
0253     862 ;       : : : : : : Read data
0253     863 ;       : : : : : : IF data error
0253     864 ;       : : : : : : : THEN
0253     865 ;       : : : : : : : Report data error
0253     866 ;       : : : : : : ENDIF
0253     867 ;       : : : : : : Write data
0253     868 ;       : : : : : : Read data
0253     869 ;       : : : : : : IF data error
0253     870 ;       : : : : : : : THEN
0253     871 ;       : : : : : : : Report data error
0253     872 ;       : : : : : : ENDIF
0253     873 ;       : : : : : : Update address
0253     874 ;       : : : : : :UNTIL all addresses done
0253     875 ;       : : : : : ENDREPEAT
0253     876 ;       : : : : : Update data
0253     877 ;       : : : : :UNTIL all data done
0253     878 ;       : : : : ENDREPEAT
0253     879 ;       : : : : Change direction
0253     880 ;       : : : :UNTIL reverse is done
0253     881 ;       : : : ENDREPEAT
0253     882 ;       : : : Update row
0253     883 ;       : : :UNTIL all rows done
0253     884 ;       : : ENDREPEAT
0253     885 ;       : : Update increment
```

```
                                       0253    886 ;          :  : UNITL increment = MSB of address
                                       0253    887 ;          :  ENDREPEAT
                                       0253    888 ;          ENDSUBTEST Moving Inversions
                                       0253    889 ;
                                       0253    890 ;
                                       0253    891 ;-
                                       0253    892
                                       0253    893
                                       0253    894            $DS_BGNSUB
                                       0253        T6_S4::
   00010030 9F    00000024'EF    FA    0253                             CALLG    $S$, @#DS$BGNSUB
   0000000C'EF    0000038D'EF    DE    025E    895            MOVAL    RENTRY1,A_RETURN ; load rentry address for unexpected MC
                                       0269    896            $DS_SETVEC_S     VECTOR=#SCB$L_MACHCHK,SRVADR=CHECK_MCHK
                          00     DD    0269                             PUSHL    #0
                  00000418'EF    DF    026B                             PUSHAL   CHECK_MCHK
                          04     DD    0271                             PUSHL    #SCB$L_MACHCHK
          00010160 9F     03     FB    0273                             CALLS    #3, @#DS$SETVEC
                  00000010'EF    D4    027A    897 TAG1:      CLRL     L_FLAGS              ; initialize flags
                  00000008'EF    D4    0280    898            CLRL     L_ROW                ; start with row 0
                  0000026E'EF    16    0286    899            JSB      CLEAR_WCS            ; initialize WCS
          00000004'EF     10     D0    028C    900 5$:        MOVL     #16,L_INC            ; initialize address increment
                  00000350'EF    16    0293    901 10$:       JSB      SETUP_ADDR           ; go set up address table for this inc
          53      00000200'EF    DE    0299    902 20$:       MOVAL    DATA_TABLE,R3        ; setup pointer to read data
          54      00000204'EF    DE    02A0    903            MOVAL    DATA_TABLE+4,R4      ; setup pointer to write data
       09 00000010'EF     03     E1    02A7    904 30$:       BBC      #V_REVERSE,L_FLAGS,40$ ; br = doing forward
          52      00000028'EF    D0    02AF    905            MOVL     WCS_MAX_SIZE_MWORDS,R2
                                       02B6    906                                          ; point offset to last address in table
                          02     11    02B6    907            BRB      50$                  ; skip
                          52     D4    02B8    908 40$:       CLRL     R2                   ; point offset to first address in table
          5B      00000000'EF    D0    02BA    909 50$:       MOVL     A_WCS_BUFFER,R11 ; get starting address of addresses
               5A      6B42      D0    02C1    910            MOVL     (R11)[R2],R10        ; load address of data
                  55     6A      D0    02C5    911            MOVL     (R10),R5             ; read data
       55     55     14  00      EF    02C8    912            EXTZV    #0,#20,R5,R5         ; extract data from longword
                  63     55      D1    02CD    913            CMPL     R5,(R3)              ; check data read
                  1D     13      02D0          914            BEQL     60$                  ; br = data good
                                       02D2    915            $DS_ERRHARD_S     UNIT=#0,MSGADR=T_MI_ERR1,PRLINK=MI_ERR,P1=R10,-
                                       02D2    916                              P2=R5,P3=(R3)
                          63     DD    02D2                             PUSHL    (R3)
                          55     DD    02D4                             PUSHL    R5
                          5A     DD    02D6                             PUSHL    R10
                  0000098E'EF    DF    02D8                             PUSHAL   MI_ERR
                  000007A0'EF    DF    02DE                             PUSHAL   T_MI_ERR1
                          00     DD    02E4                             PUSHL    #0
                          01     DD    02E6                             PUSHL    #$ER
                                       02E8        ;;; TEST 6, SUBTEST 4, ERROR 1
          000100D0 9F     07     FB    02E8                             CALLS    #$$M, @#DS$ERRHARD
                  6A     64      D0    02EF    917 60$:       MOVL     (R4),(R10)           ; write data
                  55     6A      D0    02F2    918            MOVL     (R10),R5             ; read data
       55     55     14  00      EF    02F5    919            EXTZV    #0,#20,R5,R5         ; extract data from longword
                  64     55      D1    02FA    920            CMPL     R5,(R4)              ; check data read
                  1D     13      02FD          921            BEQL     70$                  ; br = data good
                                       02FF    922            $DS_ERRHARD_S     UNIT=#0,MSGADR=T_MI_ERR2,PRLINK=MI_ERR,P1=R10,-
                                       02FF    923                              P2=R5,P3=(R4)
                          64     DD    02FF                             PUSHL    (R4)
                          55     DD    0301                             PUSHL    R5
                          5A   DD  0303                        PUSHL    R10
                 0000098E'EF   DF  0305                        PUSHAL   MI_ERR
                 000007D5'EF   DF  030B                        PUSHAL   T_MI_ERR2
                          00   DD  0311                        PUSHL    #0
                          02   DD  0313                        PUSHL    #$ER
                               0315            ;;; TEST 6, SUBTEST 4, ERROR 2
            000100D0 9F   07   FB  0315                        CALLS    #$$M, @#DS$ERRHARD
                               031C    924 70$:        $DS_BREAK
            00010058 9F   6E   FA  031C                        CALLG    (SP), @#DS$BREAK
         07 00000010'EF   03   E1  0323    925         BBC     #V_REVERSE,L_FLAGS,80$ ; br = forward direction
                     52   01   C2  032B    926         SUBL2   #1,R2               ; index to next address (reverse)
                          8A   18  032E    927         BGEQ    50$                 ; br = not finished
                          0F   11  0330    928         BRB     100$                ; skip
                     52   01   C0  0332    929 80$:    ADDL2   #1,R2               ; index to next address (forward)
            00000028'EF   52   D1  0335    930         CMPL    R2,WCS_MAX_SIZE_MWORDS
                               033C    931                                     ; finished forward
                          03   14  033C    932         BGTR    100$                ; br = finished
                        FF79   31  033E    933         BRW     50$                 ; not finished
                     84   83   D1  0341    934 100$:   CMPL    (R3)+,(R4)+         ; update data
                          63   D5  0344    935         TSTL    (R3)                ; check for data done
                          03   13  0346    936         BEQL    105$                ; br = do reverse
                        FF5C   31  0348    937         BRW     30$                 ; br = do forward for next data
         0A 00000010'EF   03   E0  034B    938 105$:   BBS     #V_REVERSE,L_FLAGS,108$ ; br = reverse done
            00000010'EF   08   C8  0353    939         BISL2   #M_REVERSE,L_FLAGS ; set reverse flag
                        FF3C   31  035A    940         BRW     20$                 ; do reverse
00000004'EF   00000004'EF   01   9C  035D    941 108$:   ROTL    #1,L_INC,L_INC    ; shift increment
  03 00000004'EF   00000020'EF   E0  0369    942         BBS     V_OVERFLOW_BIT,L_INC,109$
                               0375    943                                     ; br = all increments done
                        FF1B   31  0375    944         BRW     10$                 ; go do another increment
            00000008'EF   04   C0  0378    945 109$:   ADDL2   #4,L_ROW            ; update row to test
                               037F    946 ;       CMPL    L_ROW,#8            ; check for finished  5.2
            0C   00000008'EF   D1  037F    947         CMPL    L_ROW,#12           ; check for finished
                          03   14  0386    948         BGTR    110$                ; br = finished
                        FF01   31  0388    949         BRW     5$                  ; br = not finished
                          1C   11  038B    950 110$:   BRB     TAG2
                               038D    951 RENTRY1:
                               038D    952         $DS_ERRHARD_S   UNIT=#0,MSGADR=T_CS_PARITY1
                          00   DD  038D                        PUSHL    #0
                 000004D5'EF   DF  038F                        PUSHAL   T_CS_PARITY1
                          00   DD  0395                        PUSHL    #0
                          03   DD  0397                        PUSHL    #$ER
                               0399            ;;; TEST 6, SUBTEST 4, ERROR 3
            000100D0 9F   04   FB  0399                        CALLS    #$$M, @#DS$ERRHARD
                               03A0    953             $DS_CKLOOP       TAG1
       00010040 9F   FED6 CF   FA  03A0                        CALLG    TAG1, @#DS$CKLOOP
                     25   00   DA  03A9    954 TAG2:   MTPR    #0,#PR750$_CADR ; Re-enable cache 5.2
                               03AC    955             $DS_CLRVEC_S     VECTOR=SCB$L_MACHCHK
                 00000004'EF   DD  03AC                        PUSHL    SCB$L_MACHCHK
            00010168 9F   01   FB  03B2                        CALLS    #1, @#DS$CLRVEC
                               03B9    956             $DS_ENDSUB
                               03B9            T6_S4_X:
     00010038 9F   00000024'EF   FA  03B9                        CALLG    $S$, @#DS$ENDSUB
                               03C4    957
                               03C4    958             $DS_ENDTEST
                     50   01   D0  03C4                        MOVL     #1, R0          ; NORMAL EXIT
```

```
                          03C7        TEST_006_X::
  00010058 9F    6E   FA  03C7                          CALLG   (SP), @#DS$BREAK
                      04  03CE                          RET                       ; RETURN TO TEST SEQUENCER
                          03CF   959
                          03CF   960         $DS_PAGE
```

```
                                     03CF               .SBTTL  TEST 7:  WCS Execution Test
                                 00000000                       .PSECT  TEST_007, PAGE, NOWRT
                                     0018
                                     0018   963 ;++
                                     0018   964 ; Test description:
                                     0018   965 ;
                                     0018   966 ;       This test checks that micro code can be executed out of writable
                                     0018   967 ;       control store.
                                     0018   968 ;
                                     0018   969 ;--
                                     0018   970
                                     0018   971         $DS_BGNTEST     <WCS,DEFAULT,ALL>
                                     0018       DATA_007:
                           00000000  0018                       .LONG   0               ; TEST ARGUMENT TABLE TERMINATOR
                                     001C       TEST_007::
                               0000  001C                       .WORD   ^M<>            ; ENTRY MASK
              51   00000000'EF   D0  001E   972         MOVL    KA_PTABLE,R1            ; get address of P table
                         36 A1   B5  0025   973         TSTW    KA$W_WCS(R1)            ; check for WCS present
                            1C   12  0028   974         BNEQ    2$                      ; br = yes test WCS
              0000FE54'EF   01   D1  002A   975         CMPL    #1,DSA$GL_PASSNO        ; ckeck for first pass
                            0D   12  0031   976         BNEQ    1$                      ; br = not first pass
                                     0033   977         $DS_PRINTF_S    FORMAT=T_NO_WCS
                   00000665'EF   9F  0033                       PUSHAB  T_NO_WCS
              000100F0 9F   01   FB  0039                       CALLS   #$$N, @#DS$PRINTF
                                     0040   978 1$:     $DS_EXIT        TEST
                       50   01   D0  0040                       MOVL    #1,R0           ; NOT AN ERROR
                          0150   31  0043                       BRW     TEST_007_X      ; EXIT TEST 7
                                     0046   979 2$:
                                     0046   980         $DS_SETVEC_S    VECTOR=#SCB$L_MACHCHK,SRVADR=CHECK_MCHK
                            00   DD  0046                       PUSHL   #0
                   00000418'EF   DF  0048                       PUSHAL  CHECK_MCHK
                            04   DD  004E                       PUSHL   #SCB$L_MACHCHK
              00010160 9F   03   FB  0050                       CALLS   #3, @#DS$SETVEC
                                     0057   981         $DS_SETVEC_S    VECTOR=#SCB$L_OPCCUS,SRVADR=XFC_SERVICE,CODE=#2
                            02   DD  0057                       PUSHL   #2
                   00000410'EF   DF  0059                       PUSHAL  XFC_SERVICE
                            14   DD  005F                       PUSHL   #SCB$L_OPCCUS
              00010160 9F   03   FB  0061                       CALLS   #3, @#DS$SETVEC
                                     0068   982
                   00000014'EF   D4  0068   983         CLRL    L_PERM_FLAGS            ; Clear test - permanent flags at start
                   0000043D'EF   16  006E   984         JSB     SIZE_WCS                ; Size up the WCS for either 1k or 2k
                   0000026E'EF   16  0074   985         JSB     CLEAR_WCS               ; Clear out all of WCS
                                     007A   986         $DS_PAGE
                                     007A   987
                                     007A   988
                               007A              .SBTTL  SUBTEST 7.1:  WCS Entry/Exit
                               007A    991 ;+
                               007A    992 ; Subtest description:
                               007A    993 ;
                               007A    994 ;       This subtest attempts to execute from WCS, an instruction which will
                               007A    995 ;       load the 'D' register with an initial value and then decrement the
                               007A    996 ;       value and write it into general purpose register R5. The XFC
                               007A    997 ;       instruction is used to execute from WCS and then the register is
                               007A    998 ;       checked for the correct results.
                               007A    999 ;
                               007A   1000 ; Subtest steps:
                               007A   1001 ;
                               007A   1002 ;       BGNSUBTEST WCS Entry/Exit
                               007A   1003 ;       :  Initialize flags
                               007A   1004 ;       :  Initialize register
                               007A   1005 ;       :  Write micro words into WCS
                               007A   1006 ;       :  Execute XFC to enter WCS
                               007A   1007 ;       :  Check register
                               007A   1008 ;       :  IF Register contents not correct
                               007A   1009 ;       :  :  THEN
                               007A   1010 ;       :  :  Report error
                               007A   1011 ;       :  ENDIF
                               007A   1012 ;       ENDSUBTEST WCS Entry/Exit
                               007A   1013 ;
                               007A   1014 ;
                               007A   1015 ;
                               007A   1016 ;-
                               007A   1017
                               007A   1018         $DS_BGNSUB
                               007A        T7_S1::
  00010030 9F   00000030'EF FA 007A                 CALLG   $S$, @#DS$BGNSUB
                00000010'EF D4 0085   1019         CLRL    L_FLAGS          ; initialize flags
                00000000'EF D4 008B   1020         CLRL    L_COUNT          ; setup count for this subtest
  0000000C'EF   000000CD'EF DE 0091   1021         MOVAL   RENTRY2,A_RETURN ; load rentry address for unexpected MC
                     55   10 D0 009C   1022 TAG3:  MOVL    #16,R5           ; initialize test register
                0000038C'EF 16 009F   1023         JSB     LOAD_MICRO       ; load wcs with micro code and enable WCS
                            FC 00A5   1024         XFC                      ; execute instruction in WCS
                            01 00A6   1025         NOP                      ; skip a location
                00000018'FF D4 00A7   1026         CLRL    @A_WCS_ADDRESS   ; disable WCS
                     0F   55 D1 00AD   1027         CMPL    R5,#15           ; check that register decremented twice
                          36 13 00B0   1028         BEQL    TAG4             ; br = data good
                               00B2   1029         $DS_ERRHARD_S   UNIT=#0,MSGADR=T_ENTRY_ERR,PRLINK=WCS_ERR,P1=R5
                          55 DD 00B2                         PUSHL   R5
                000009B4'EF DF 00B4                         PUSHAL  WCS_ERR
                0000080A'EF DF 00BA                         PUSHAL  T_ENTRY_ERR
                          00 DD 00C0                         PUSHL   #0
                          01 DD 00C2                         PUSHL   #$ER
                               00C4        ;;; TEST 7, SUBTEST 1, ERROR 1
        000100D0 9F       05 FB 00C4                         CALLS   #$$M, @#DS$ERRHARD
                          1B 11 00CB   1030         BRB     TAG4
                               00CD   1031 RENTRY2:
                               00CD   1032         $DS_ERRHARD_S   UNIT=#0,MSGADR=T_CS_PARITY2
                          00 DD 00CD                         PUSHL   #0
                00000508'EF DF 00CF                         PUSHAL  T_CS_PARITY2
                          00 DD 00D5                         PUSHL   #0
                     02   DD  00D7                  PUSHL   #$ER
                              00D9       ;;; TEST 7, SUBTEST 1, ERROR 2
        000100D0 9F    04   FB  00D9                  CALLS   #$$M, a#DS$ERRHARD
                              00E0  1033         $DS_CKLOOP      TAG3
     00010040 9F    B9 AF   FA  00E0                  CALLG   TAG3, a#DS$CKLOOP
                              00E8  1034 TAG4:
             0000026E'EF   16  00E8  1035         JSB     CLEAR_WCS       ; Clear/Disable out all of WCS
                              00EE  1036         $DS_ENDSUB
                              00EE       T7_S1_X:
  00010038 9F   00000030'EF   FA  00EE                  CALLG   $S$, a#DS$ENDSUB
                              00F9  1037
                              00F9  1038         $DS_PAGE
                              00F9  1039
                              00F9  1040
```

```
                                     00F9                  .SBTTL  SUBTEST 7.2:  WCS 'Next Field  Addressing
                                     00F9  1043 ;+
                                     00F9  1044 ; Subtest description:
                                     00F9  1045 ;
                                     00F9  1046 ;          This subtest checks the WCS "next field' portion of the micro word to
                                     00F9  1047 ;          insure that each bit produces the proper transfer address. This is
                                     00F9  1048 ;          accomplished by filling WCS memory with micro words that decrement
                                     00F9  1049 ;          general purpose register R5 and have a  next field' which points to
                                     00F9  1050 ;          the next WCS location. This subtest is an extened version of the
                                     00F9  1051 ;          previous subtest and runs thru all of WCS memory. An XFC instruction
                                     00F9  1052 ;          is executed to execute from WCS and afterwards R5 is checked to have
                                     00F9  1053 ;          the expected value.
                                     00F9  1054 ;
                                     00F9  1055 ; Subtest steps:
                                     00F9  1056 ;
                                     00F9  1057 ;          BGNSUBTEST WCS 'Next Field  Addressing
                                     00F9  1058 ;          :  Initialize flags
                                     00F9  1059 ;          :  Initialize register
                                     00F9  1060 ;          :  Write WCS with micro words for all WCS
                                     00F9  1061 ;          :  Execute XFC instruction
                                     00F9  1062 ;          :  Check register to have decremented properly
                                     00F9  1063 ;          :  IF register contains wrong data
                                     00F9  1064 ;          :  :  THEN
                                     00F9  1065 ;          :  :  Report error
                                     00F9  1066 ;          :  ENDIF
                                     00F9  1067 ;          ENDSUBTEST WCS  Next Field' Addressing
                                     00F9  1068 ;
                                     00F9  1069 ;--
                                     00F9  1070
                                     00F9  1071         $DS_BGNSUB
                                     00F9       T7_S2::
     00010030 9F   0000003C'EF   FA  00F9                       CALLG   $S$, a#DS$BGNSUB
                   00000010'EF   D4  0104  1072         CLRL    L_FLAGS         ; initialize flags
     00000000'EF   00000030'EF   D0  010A  1073         MOVL    MICRO_WORD_COUNT,L_COUNT
                                     0115  1074                                 ; setup count for all locations
     0000000C'EF   00000155'EF   DE  0115  1075         MOVAL   RENTRY3,A_RETURN ; load rentry address for unexpected MC
              55   00000034'EF   D0  0120  1076 TAG5:   MOVL    TEST_VALUE,R5   ; initialize test register
                   0000038C'EF   16  0127  1077         JSB     LOAD_MICRO      ; load wcs with micro code
                                 FC  012D  1078         XFC                     ; execute instruction in WCS
                                 01  012E  1079         NOP                     ; skip a location
                   00000018'FF   D4  012F  1080         CLRL    aA_WCS_ADDRESS  ; disable WCS
                       0F   55   D1  0135  1081         CMPL    R5,#15          ; check that register decremented
                            36   13  0138  1082         BEQL    TAG6            ; br = data good
                                     013A  1083         $DS_ERRHARD_S   UNIT=#0,MSGADR=T_NXFIELD_ERR,PRLINK=WCS_ERR,P1=R5
                            55   DD  013A                       PUSHL   R5
                   000009B4'EF   DF  013C                       PUSHAL  WCS_ERR
                   0000082F'EF   DF  0142                       PUSHAL  T_NXFIELD_ERR
                            00   DD  0148                       PUSHL   #0
                            01   DD  014A                       PUSHL   #$ER
                                     014C       ;;; TEST 7, SUBTEST 2, ERROR 1
              000100D0 9F   05   FB  014C                       CALLS   #$$M, a#DS$ERRHARD
                            1B   11  0153  1084         BRB     TAG6
                                     0155  1085 RENTRY3:
                                     0155  1086         $DS_ERRHARD_S   UNIT=#0,MSGADR=T_CS_PARITY3
                            00   DD  0155                       PUSHL   #0
                   00000534'EF   DF  0157                       PUSHAL  T_CS_PARITY3
                            00   DD  015D                       PUSHL   #0
                            02   DD  015F                       PUSHL   #$ER
                                     0161       ;;; TEST 7, SUBTEST 2, ERROR 2
              000100D0 9F   04   FB  0161                       CALLS   #$$M, @#DS$ERRHARD
                                     0168  1087         $DS_CKLOOP      TAG5
           00010040 9F    B5 AF   FA  0168                       CALLG   TAG5, @#DS$CKLOOP
                                     0170  1088 TAG6:
                   0000026E'EF   16  0170  1089         JSB     CLEAR_WCS       ; Clear/Disable out all of WCS before exiting
                                     0176  1090                                 ; this test
                                     0176  1091         $DS_ENDSUB
                                     0176       T7_S2_X:
     00010038 9F   0000003C'EF   FA  0176                       CALLG   $S$, @#DS$ENDSUB
                                     0181  1092
                                     0181  1093         $DS_CLRVEC_S    VECTOR=#SCB$L_MACHCHK
                            04   DD  0181                       PUSHL   #SCB$L_MACHCHK
              00010168 9F   01   FB  0183                       CALLS   #1, @#DS$CLRVEC
                                     018A  1094         $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                            14   DD  018A                       PUSHL   #SCB$L_OPCCUS
              00010168 9F   01   FB  018C                       CALLS   #1, @#DS$CLRVEC
                                     0193  1095         $DS_ENDTEST
                       50   01   D0  0193                       MOVL    #1, R0          ; NORMAL EXIT
                                     0196       TEST_007_X::
              00010058 9F   6E   FA  0196                       CALLG   (SP), @#DS$BREAK
                                 04  019D                       RET                     ; RETURN TO TEST SEQUENCER
                                     019E  1096
                                     019E  1097         $DS_ENDMOD
                                 00000000                       .PSECT  $TSTCNT, NOEXE, NOWRT, OVR, LONG
                           00000007  0000                       .LONG   $TN
                                     0004  1098         .END
```

```
$$$                              = 0000003C R      05        DS$ENDPASS                        00010010
$$ARGS                           = 0000000A                  DS$ENDSUB                         00010038
$$E                              = 00000001                  DS$ERRDEV                         000100C8
$$M                              = 00000004                  DS$ERRHARD                        000100D0
$$N                              = 00000000                  DS$ERRPREP                        00010108
$$S                              = FFFFFFFF                  DS$ERRSOFT                        000100D8
$$T1                             = 0000002C                  DS$ERRSYS                         000100C0
$ENV                             = 00000001  G               DS$ESCAPE                         00010050
$ER                              = FFFFFFFF                  DS$FREEDBGSYM                     000101B8
$MO                              = 00000001  G               DS$GETBUF                         00010120
$S$                              = 0000003C R      05        DS$GETMEM                         00010130
$ST                              = 00000000                  DS$GETTERM                        000101C8
$TN                              = 00000007                  DS$GPHARD                         00010018
ADDR_ERR                           00000938 R      01        DS$HELP                           000101B0
ADDR_SIZE                          00000024 R      01        DS$INITSCB                        00010170
ALL                              = 00000009  G               DS$INLOOP                         00010048
A_ENA_REGISTER                     00000038 R      01        DS$K_ERROR                      = 00000002
A_RETURN                           0000000C R      01        DS$K_NORMAL                     = 00000001
A_WCS_ADDRESS                      00000018 R      01        DS$K_SEVERE                     = 00000004
A_WCS_BUF                          00000044 R      01        DS$K_SUBSYS                     = 00000066
A_WCS_BUFFER                       ********   X    01        DS$K_WARNING                    = 00000000
A_WCS_ENA_ADDR                     0000001C R      01        DS$LOAD                           00010198
BASE_006                           00000000 R      03        DS$LOADPCS                        000101C0
BASE_007                           00000000 R      06        DS$MAPDBGBLOCK                    00010118
BIT...                           = 0000001A                  DS$MEMSIZE                        000101D8
CEP_FUNCTIONAL                   = 00000000                  DS$MMOFF                          00010158
CEP_REPAIR                       = 00000001                  DS$MMON                           00010150
CHECK_MCHK                         00000418 R      01        DS$MOVPHY                         00010148
CLEAR_BUFFER                       00000258 R      01        DS$MOVVRT                         00010140
CLEAR_WCS                          0000026E R      01        DS$PARSE                          000100B8
CODEVECTORS                        000001C4 R      01        DS$PRINTB                         000100E0
CODE_WORDS                         000001D4 R      01        DS$PRINTF                         000100F0
DATA_006                           00000014 R      03        DS$PRINTS                         000100F8
DATA_007                           00000018 R      06        DS$PRINTSIG                       00010100
DATA_PATTERN                       0000003C R      01        DS$PRINTX                         000100E8
DATA_TABLE                         00000200 R      01        DS$PROBE                          000101A0
DECODED_WORDS                      000000C4 R      01        DS$RELBUF                         00010128
DEFAULT                          = 00000000  G               DS$RELMEM                         00010138
DPERR                              000008C5 R      01        DS$SERVICE_DISPATCHER             000101D0
DS$ABORT                           00010020                  DS$SETIPL                         00010178
DS$ASKADR                          00010090                  DS$SETMAP                         00010188
DS$ASKDATA                         00010080                  DS$SETPRIEXV                      00010110
DS$ASKLGCL                         00010098                  DS$SETVEC                         00010160
DS$ASKSTR                          000100A0                  DS$SHOCHAN                        00010190
DS$ASKVLD                          00010088                  DS$SUMMARY                        00010028
DS$ATTACH                          000101A8                  DS$WAITMS                         00010060
DS$BGNSUB                          00010030                  DS$WAITUS                         00010068
DS$BRANCH                          000100A8                  DS$_ARITH                       = 006600D0
DS$BREAK                           00010058                  DS$_ASBE                        = 00660118
DS$CANWAIT                         00010070                  DS$_BADLINK                     = 006600F0
DS$CHANNEL                         00010180                  DS$_BADTYPE                     = 006600E8
DS$CKLOOP                          00010040                  DS$_BIIC                        = 00660120
DS$CLRVEC                          00010168                  DS$_CHME                        = 006600A8
DS$CNTRLC                          00010078                  DS$_CHMK                        = 006600E0
DS$CVTREG                          000100B0                  DS$_DEVNAME                     = 00660108
```

```
DS$_ERROR                      = 00660002                 ENV$S_LEVEL                    = 00000001
DS$_FHWE                       = 00660068                 ENV$S_SUPER                    = 00000008
DS$_FRAGBUF                    = 00660080                 ENV$V_DOMAIN                   = 00000001
DS$_ICBUSY                     = 006600C8                 ENV$V_LEVEL                    = 00000000
DS$_ICERR                      = 006600C0                 ENV$V_SUPER                    = 00000002
DS$_IHWE                       = 00660060                 ENV$_CPU                       = 00000000
DS$_ILLCHAR                    = 00660018                 ENV$_FUNCTIONAL                = 00000000
DS$_ILLPAGCNT                  = 00660078                 ENV$_REPAIR                    = 00000001
DS$_ILLUNIT                    = 00660100                 ENV$_SUPER                     = 00000001
DS$_INSFMEM                    = 00660050                 ENV$_SYSTEM                    = 00000001
DS$_IPL2HI                     = 006600B8                 ERR$_MSGADR                    = 0000000C
DS$_IVADDR                     = 00660040                 ERR$_NARGS                     = 0000000A
DS$_IVVECT                     = 00660038                 ERR$_NUM                       = 00000004
DS$_KRNLSTK                    = 00660090                 ERR$_P1                        = 00000014
DS$_LOGIC                      = 00660070                 ERR$_P2                        = 00000018
DS$_MCHK                       = 00660088                 ERR$_P3                        = 0000001C
DS$_MMOFF                      = 00660058                 ERR$_P4                        = 00000020
DS$_NEEDUNIT                   = 006600F8                 ERR$_P5                        = 00000024
DS$_NODE                       = 00660128                 ERR$_P6                        = 00000028
DS$_NOPCS                      = 00660110                 ERR$_POINTER                   = 00000010
DS$_NORMAL                     = 00660001                 ERR$_UNIT                      = 00000008
DS$_NOSUPPORT                  = 006600B1                 F_ROWERR1                        00000742 R     01
DS$_NOTDON                     = 00660030                 F_ROWERR2                        0000075D R     01
DS$_NOTIMP                     = 006600B0                 F_WCS_ERR                        0000085D R     01
DS$_NULLSTR                    = 00660010                 HALT                           = 00000004  G
DS$_OVERFLOW                   = 00660008                 HP$A_DEPENDENT                   00000032
DS$_POWER                      = 00660098                 HP$A_DEVICE                      00000018
DS$_PROGERR                    = 00660020                 HP$A_DVA                         0000001C
DS$_SEVERE                     = 00660004                 HP$A_LINK                        00000020
DS$_TRANSL                     = 006600A0                 HP$B_DRIVE                       0000000B
DS$_TRUNCATE                   = 00660028                 HP$B_FLAGS                       0000000A
DS$_UNEXPINT                   = 006600D8                 HP$Q_DEVICE                      00000000
DS$_VASFULL                    = 00660048                 HP$T_DEVICE                      0000000C
DS$_WARNING                    = 00660000                 HP$T_TYPE                        00000026
DSA$AL_APTMAIL                   0000FE00                 HP$W_SIZE                        00000008
DSA$AT_APTTXT                    0000FA00                 HP$W_VECTOR                      00000024
DSA$GL_APTCOM                    0000FE04                 IPR                            = 00000006  G
DSA$GL_DEVLEN                    0000FE58                 KA$B_ACC_TYPE                    00000042  G
DSA$GL_ERRNO                     0000FE44                 KA$B_NODE_ID                     00000046  G
DSA$GL_EVENT                     0000FE48                 KA$B_RESERVED                    00000043  G
DSA$GL_FLAGS                     0000FE00                 KA$B_SID_TYPE                    0000003D  G
DSA$GL_MSGTYP                    0000FE40                 KA$K_LEN                         0000004B  G
DSA$GL_PASSES                    0000FE08                 KA$L_FLAGS                       00000032  G
DSA$GL_PASSNO                    0000FE54                 KA$L_MEM_SIZE                    00000047  G
DSA$GL_SECTNO                    0000FE10                 KA$L_SID                         0000003A  G
DSA$GL_SID                       0000FE14                 KA$M_CHAR                      = 00000100  G
DSA$GL_SUBTNO                    0000FE4C                 KA$M_CRC                       = 00000010  G
DSA$GL_TESTNO                    0000FE50                 KA$M_D_FLOAT                   = 00000002  G
DSA$GL_UNITS                     0000FE0C                 KA$M_EDIT                      = 00000080  G
DSA$GQ_MSGPTR                    0000FE68                 KA$M_F_FLOAT                   = 00000001  G
DSA$GT_DEVNAM                    0000FE5C                 KA$M_G_FLOAT                   = 00000004  G
ENV$M_DOMAIN                   = 00000002                 KA$M_H_FLOAT                   = 00000008  G
ENV$M_LEVEL                    = 00000001                 KA$M_PACKED                    = 00000020  G
ENV$M_SUPER                    = 000003FC                 KA$M_PACK_MUL                  = 00000040  G
ENV$S_DOMAIN                   = 00000001                 KA$M_SB_ERR                    = 02000000  G
```

```
KA$M_TODR                        = 00010000  G                PAR$_OCT                       = 00000008
KA$M_WCS_LD                      = 01000000  G                PAR$_YES                       = 00000001
KA$V_CHAR                        = 00000008  G                PCS                            = 00000008  G
KA$V_CRC                         = 00000004  G                POWER_FAIL                     = 00000003  G
KA$V_D_FLOAT                     = 00000001  G                PR$_SID                          ********   X     01
KA$V_EDIT                        = 00000007  G                PR750$_CADR                    = 00000025
KA$V_F_FLOAT                     = 00000000  G                PSL$K_EXEC                     = 00000001
KA$V_G_FLOAT                     = 00000002  G                PSL$K_KERNEL                   = 00000000
KA$V_H_FLOAT                     = 00000003  G                PSL$K_SUPER                    = 00000002
KA$V_PACKED                      = 00000005  G                PSL$K_USER                     = 00000003
KA$V_PACK_MUL                    = 00000006  G                PSL$M_CBIT                     = 00000001
KA$V_SB_ERR                      = 00000019  G                PSL$M_NBIT                     = 00000008
KA$V_TODR                        = 00000010  G                PSL$M_VBIT                     = 00000002
KA$V_WCS_LD                      = 00000018  G                PSL$M_ZBIT                     = 00000004
KA$W_LATENCY                       0000003E  G                PSL$V_CBIT                     = 00000000
KA$W_MEM_SIZE                      00000044  G                PSL$V_NBIT                     = 00000003
KA$W_PROGRESS                      00000040  G                PSL$V_VBIT                     = 00000001
KA$W_RESERVED                      00000038  G                PSL$V_ZBIT                     = 00000002
KA$W_WCS                           00000036  G                RENTRY1                          0000038D R       03
KA_PTABLE                          ********   X     01        RENTRY2                          000000CD R       06
LOAD_MICRO                         0000038C R       01        RENTRY3                          00000155 R       06
L_COUNT                            00000000 R       01        RM_DECODE                        0000028F R       01
L_FLAGS                            00000010 R       01        ROWERR                           000008E8 R       01
L_INC                              00000004 R       01        SCB$L_ACCESS                     00000020
L_PERM_FLAGS                       00000014 R       01        SCB$L_ARITH                      00000034
L_REV_CHECK                        ********   X     01        SCB$L_BREAK                      0000002C
L_ROW                              00000008 R       01        SCB$L_CHME                       00000044
MANUAL                           = 00000001  G                SCB$L_CHMK                       00000040
MCHK                             = 00000005  G                SCB$L_CHMS                       00000048
MICRO_DATA                         00000224 R       01        SCB$L_CHMU                       0000004C
MICRO_WORD_COUNT                   00000030 R       01        SCB$L_COMPAT                     00000030
MI_ERR                             0000098E R       01        SCB$L_KNLSTK                     00000008
MSG_006                            00000002 R       03        SCB$L_MACHCHK                    00000004
MSG_007                            00000002 R       06        SCB$L_OPCCUS                     00000014
M_BIT19                          = 00080000                   SCB$L_OPCDEC                     00000010
M_BIT20                          = 00100000                   SCB$L_POWER                      0000000C
M_DATA_ERR                       = 00000002                   SCB$L_RADRMOD                    0000001C
M_OLD_WCS                        = 00000001                   SCB$L_ROPRAND                    00000018
M_PAT                            = 00000001                   SCB$L_RXDB                       000000F8
M_REVERSE                        = 00000008                   SCB$L_SFTLVL1                    00000084
M_ROW_ERR                        = 00000004                   SCB$L_SFTLVL10                   000000A8
M_XFC_ERR                        = 00000010                   SCB$L_SFTLVL11                   000000AC
NOISY_WORDS                        00000144 R       01        SCB$L_SFTLVL12                   000000B0
PAR$M_ATDEF                      = 00000010                   SCB$L_SFTLVL13                   000000B4
PAR$M_ATHI                       = 00000008                   SCB$L_SFTLVL14                   000000B8
PAR$M_ATLO                       = 00000004                   SCB$L_SFTLVL15                   000000BC
PAR$M_NODEF                      = 00000002                   SCB$L_SFTLVL2                    00000088
PAR$V_ATDEF                      = 00000004                   SCB$L_SFTLVL3                    0000008C
PAR$V_ATHI                       = 00000003                   SCB$L_SFTLVL4                    00000090
PAR$V_ATLO                       = 00000002                   SCB$L_SFTLVL5                    00000094
PAR$V_NODEF                      = 00000001                   SCB$L_SFTLVL6                    00000098
PAR$_BIN                         = 00000002                   SCB$L_SFTLVL7                    0000009C
PAR$_DEC                         = 0000000A                   SCB$L_SFTLVL8                    000000A0
PAR$_HEX                         = 00000010                   SCB$L_SFTLVL9                    000000A4
PAR$_NO                          = 00000000                   SCB$L_TBIT                       00000028
```

```
SCB$L_TIMER                     000000C0                T7_S1_X                         000000EE R      06
SCB$L_TRANSL                    00000024                T7_S2                           000000F9 RG     06
SCB$L_TXDB                      000000FC                T7_S2_X                         00000176 R      06
SCB$L_ZERO                      00000000                TAG1                            0000027A R      03
SEP_FUNCTIONAL                = 00000002                TAG2                            000003A9 R      03
SEP_REPAIR                    = 00000003                TAG3                            0000009C R      06
SETUP_ADDR                      00000350 R      01      TAG4                            000000E8 R      06
SIZ...                        = 00000001                TAG5                            00000120 R      06
SIZE_WCS                        0000043D R      01      TAG6                            00000170 R      06
SYS$ALLOC                       00010238                TEST_006                        00000018 RG     03
SYS$ASCTIM                      00010248                TEST_006_X                      000003C7 RG     03
SYS$ASSIGN                      00010250                TEST_007                        0000001C RG     06
SYS$BINTIM                      00010258                TEST_007_X                      00000196 RG     06
SYS$CANCEL                      00010260                TEST_VALUE                      00000034 R      01
SYS$CANTIM                      00010268                TU58                          = 00000002  G
SYS$CLOSE                       000105B8                T_ADR_ERR                       00000772 R      01
SYS$CLREF                       00010298                T_BAD_ATTACH                    0000056B R      01
SYS$CONNECT                     000105C0                T_BAD_ATTACH2                   000005DB R      01
SYS$DALLOC                      000102D8                T_CS_PARITY1                    000004D5 R      01
SYS$DASSGN                      000102E0                T_CS_PARITY2                    00000508 R      01
SYS$DISCONNECT                  000105D0                T_CS_PARITY3                    00000534 R      01
SYS$FAO                         00010350                T_DATAPATH                      00000691 R      01
SYS$FAOL                        00010358                T_DPERR                         000006B7 R      01
SYS$GET                         00010580                T_ENTRY_ERR                     0000080A R      01
SYS$GETCHN                      000104C8                T_MI_ERR1                       000007A0 R      01
SYS$GETTIM                      00010378                T_MI_ERR2                       000007D5 R      01
SYS$LKWSET                      000103A0                T_NO_RESPONCE                   00000634 R      01
SYS$NUMTIM                      000103B8                T_NO_WCS                        00000665 R      01
SYS$OPEN                        00010608                T_NXFIELD_ERR                   0000082F R      01
SYS$QIO                         000103C8                T_ROW                           000006F0 R      01
SYS$QIOW                        00010200                T_ROWERR                        00000717 R      01
SYS$READ                        00010590                T_XFC_ERR                       00000891 R      01
SYS$READEF                      000103D0                V_BIT14                       = 0000000E
SYS$SETAST                      000103F8                V_BIT15                       = 0000000F
SYS$SETEF                       00010400                V_DATA_ERR                    = 00000001
SYS$SETIMR                      00010420                V_OLD_WCS                     = 00000000
SYS$SETPRI                      00010428                V_OVERFLOW_BIT                  00000020 R      01
SYS$SETPRT                      00010430                V_PAT                         = 00000000
SYS$SETRWM                      00010438                V_REVERSE                     = 00000003
SYS$SETSWM                      00010448                V_ROW_ERR                     = 00000002
SYS$ULKPAG                      00010460                V_XFC_ERR                     = 00000004
SYS$ULWSET                      00010468                WCS                           = 00000007  G
SYS$UNWIND                      00010470                WCS_ERR                         000009B4 R      01
SYS$WAITFR                      00010478                WCS_MAX_SIZE_LWORDS             0000002C R      01
SYS$WFLAND                      00010488                WCS_MAX_SIZE_MWORDS             00000028 R      01
SYS$WFLOR                       00010490                XFC_SERVICE                     00000410 R      01
T6_S1                           00000094 RG     03
T6_S1_X                         00000120 R      03
T6_S2                           0000012B RG     03
T6_S2_X                         000001B2 R      03
T6_S3                           000001BD RG     03
T6_S3 X                         00000248 R      03
T6_S4                           00000253 RG     03
T6_S4_X                         000003B9 R      03
T7_S1                           0000007A RG     06
```

```
                                              +----------------+
                                              ! Psect synopsis !
                                              +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (   0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
ECKAXE                          000009E5  (  2533.)  01 (   1.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC PAGE
$ABS$                           00010610  (67088.)  02 (   2.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
TEST_006                        000003CF  (   975.)  03 (   3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC PAGE
DISPATCH                        00000030  (    48.)  04 (   4.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD   NOWRT NOVEC LONG
ARGLIST                         00000048  (    72.)  05 (   5.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC LONG
TEST_007                        0000019E  (   414.)  06 (   6.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC PAGE
$TSTCNT                         00000004  (     4.)  07 (   7.)  NOPIC   USR   OVR   REL   LCL NOSHR NOEXE   RD   NOWRT NOVEC LONG
```

```
                                         +-------------------------+
                                         ! Symbol Cross Reference !
                                         +-------------------------+

SYMBOL                           VALUE         DEFINITION        REFERENCES...
-------                          -----         ----------        -------------
$$$                              =0000003C-R   1071    (15)       1018    (14)      1071    (15)      669     (9)       737     (10)
                                                                  803     (11)      894     (12)
$$ARGS                           =0000000A     85      (2)        85      (2)
$$E                              =00000001     1071    (15)       1036    (14)      1091    (15)      694     (9)       762     (10)
                                                                  827     (11)      956     (12)
$$M                              =00000004     1086    (15)       1029    (14)      1032    (14)      1083    (15)      1086    (15)
                                                                  597     (8)       610     (8)       683     (9)       760     (10)
                                                                  824     (11)      916     (12)      923     (12)      952     (12)
                                                                  971     (13)
$$N                              =00000000     1086    (15)       1029    (14)      1032    (14)      1083    (15)      1086    (15)
                                                                #-430     (3)     #-432     (3)       525     (7)     #-530     (7)
                                                                  536     (7)       538     (7)     #-546     (7)       554     (7)
                                                                  556     (7)       568     (7)       576     (7)     #-578     (7)
                                                                #-603     (8)       610     (8)       683     (9)       760     (10)
                                                                  824     (11)      89      (2)       916     (12)      923     (12)
                                                                  952     (12)    #-977     (13)
$$S                              =FFFFFFFF     1095    (15)       1018    (14)      1042    (15)      1071    (15)      585     (8)
                                                                  597     (8)       620     (9)       669     (9)       698     (10)
                                                                  737     (10)      766     (11)      803     (11)      831     (12)
                                                                  894     (12)      962     (13)      971     (13)      990     (14)
$$T1                             =0000002C     85      (2)        85      (2)
$ENV                             =00000001     68      (2)
$ER                              =FFFFFFFF     1095    (15)       1018    (14)    #-1029    (14)    #-1032    (14)      1071    (15)
                                                                #-1083    (15)    #-1086    (15)    #-610     (8)       669     (9)
                                                                #-683     (9)       737     (10)    #-760     (10)      803     (11)
                                                                #-824     (11)      894     (12)    #-916     (12)    #-923     (12)
                                                                #-952     (12)
$MO                              =00000001     1097    (15)       1097    (15)
$S$                              =0000003C-R   1071    (15)       1018    (14)      1036    (14)      1071    (15)      1091    (15)
                                                                  669     (9)       694     (9)       737     (10)      762     (10)
                                                                  803     (11)      827     (11)      894     (12)      956     (12)
$ST                              =00000000     1095    (15)       1018    (14)      1029    (14)      1032    (14)      1033    (14)
                                                                  1036    (14)      1042    (15)      1071    (15)      1083    (15)
                                                                  1086    (15)      1087    (15)      1091    (15)      1095    (15)
                                                                  597     (8)       610     (8)       611     (8)       620     (9)
                                                                  669     (9)       683     (9)       684     (9)       694     (9)
                                                                  698     (10)      737     (10)      760     (10)      762     (10)
                                                                  766     (11)      803     (11)      824     (11)      827     (11)
                                                                  831     (12)      894     (12)      916     (12)      923     (12)
                                                                  952     (12)      953     (12)      956     (12)      958     (12)
                                                                  971     (13)      990     (14)
$TN                              =00000007     1097    (15)       1029    (14)      1032    (14)      1042    (15)      1083    (15)
                                                                  1086    (15)      1095    (15)      1097    (15)      431     (3)
                                                                  433     (3)       585     (8)       597     (8)       604     (8)
                                                                  610     (8)       620     (9)       683     (9)       698     (10)
                                                                  760     (10)      766     (11)      824     (11)      831     (12)
                                                                  916     (12)      923     (12)      952     (12)      958     (12)
                                                                  962     (13)      971     (13)      978     (13)      990     (14)
```

```
ADDR_ERR                          00000938-R   544     (7)       824      (11)
ADDR_SIZE                         00000024-R   132     (2)     #-326      (3)     #-441      (3)
ALL                              =00000009     89      (2)       597      (8)       971      (13)
A_ENA_REGISTER                    00000038-R   150     (2)     #-442      (3)
A_RETURN                          0000000C-R   99      (2)     #-1021     (14)    #-1075     (15)    #-398      (3)     #-895      (12)
A_WCS_ADDRESS                     00000018-R   124     (2)     #-1026     (14)    #-1080     (15)    #-239      (3)     #-319      (3)
                                                               #-342      (3)     #-373      (3)     #-532      (7)     #-607      (8)
                                                               #-616      (8)     #-671      (9)     #-740      (10)    #-745      (10)
                                                               #-806      (11)
A_WCS_BUF                         00000044-R   158     (2)       228      (3)       533      (7)       747      (10)      754      (10)
A_WCS_BUFFER                      00000000-XR                  #-318      (3)     #-909      (12)
A_WCS_ENA_ADDR                    0000001C-R   126     (2)     #-238      (3)     #-340      (3)
BASE_006                          00000000-R   585     (8)       597      (8)
BASE_007                          00000000-R   962     (13)      971      (13)
BIT...                           =0000001A     84      (2)       68       (2)       82       (2)       83       (2)       84       (2)
CEP_FUNCTIONAL                   =00000000     68      (2)
CEP_REPAIR                       =00000001     68      (2)       68       (2)
CHECK_MCHK                        00000418-R   390     (3)       896      (12)      980      (13)
CLEAR_BUFFER                      00000258-R   227     (3)       739      (10)
CLEAR_WCS                         0000026E-R   236     (3)       1035     (14)      1089     (15)      615      (8)       899      (12)
                                                                 985      (13)
CODEVECTORS                       000001C4-R   164     (2)     #-295      (3)
CODE_WORDS                        000001D4-R   169     (2)       809      (11)
DATA_006                          00000014-R   597     (8)       597      (8)
DATA_007                          00000018-R   971     (13)      971      (13)
DATA_PATTERN                      0000003C-R   154     (2)     #-673      (9)     #-688      (9)
DATA_TABLE                        00000200-R   181     (2)       902      (12)      903      (12)
DECODED_WORDS                     000000C4-R   160     (2)     #-255      (3)       285      (3)     #-293      (3)       305      (3)
                                                                 550      (7)       818      (11)
DEFAULT                          =00000000     89      (2)       597      (8)       971      (13)
DPERR                             000008C5-R   520     (7)       683      (9)
DS$ABORT                          00010020                       396      (3)
DS$BGNSUB                         00010030                       1018     (14)      1071     (15)      669      (9)       737      (10)
                                                                 803      (11)      894      (12)
DS$BREAK                          00010058                       1095     (15)      924      (12)      958      (12)
DS$CKLOOP                         00010040                       1033     (14)      1087     (15)      611      (8)       684      (9)
                                                                 953      (12)
DS$CLRVEC                         00010168                       1093     (15)      1094     (15)      955      (12)
DS$ENDSUB                         00010038                       1036     (14)      1091     (15)      694      (9)       762      (10)
                                                                 827      (11)      956      (12)
DS$ERRHARD                        000100D0                       1029     (14)      1032     (14)      1083     (15)      1086     (15)
                                                                 610      (8)       683      (9)       760      (10)      824      (11)
                                                                 916      (12)      923      (12)      952      (12)
DS$K_ERROR                       =00000002     82      (2)
DS$K_NORMAL                      =00000001     82      (2)
DS$K_SEVERE                      =00000004     82      (2)
DS$K_SUBSYS                      =00000066     82      (2)       82       (2)
DS$K_WARNING                     =00000000     82      (2)
DS$PRINTB                         000100E0                       525      (7)       530      (7)       536      (7)       546      (7)
                                                                 554      (7)       568      (7)       576      (7)       578      (7)
DS$PRINTF                         000100F0                       430      (3)       432      (3)       603      (8)       977      (13)
DS$PRINTSIG                       00010100                       395      (3)
DS$PRINTX                         000100E8                       538      (7)       556      (7)
DS$PROBE                          000101A0                       608      (8)
DS$SETVEC                         00010160                       896      (12)      980      (13)      981      (13)
```

```
DS$_ARITH                      =006600D0     82     (2)
DS$_ASBE                       =00660118     82     (2)
DS$_BADLINK                    =006600F0     82     (2)
DS$_BADTYPE                    =006600E8     82     (2)
DS$_BIIC                       =00660120     82     (2)
DS$_CHME                       =006600A8     82     (2)
DS$_CHMK                       =006600E0     82     (2)
DS$_DEVNAME                    =00660108     82     (2)
DS$_ERROR                      =00660002     82     (2)
DS$_FHWE                       =00660068     82     (2)
DS$_FRAGBUF                    =00660080     82     (2)
DS$_ICBUSY                     =006600C8     82     (2)
DS$_ICERR                      =006600C0     82     (2)
DS$_IHWE                       =00660060     82     (2)
DS$_ILLCHAR                    =00660018     82     (2)
DS$_ILLPAGCNT                  =00660078     82     (2)
DS$_ILLUNIT                    =00660100     82     (2)
DS$_INSFMEM                    =00660050     82     (2)
DS$_IPL2HI                     =006600B8     82     (2)
DS$_IVADDR                     =00660040     82     (2)
DS$_IVVECT                     =00660038     82     (2)
DS$_KRNLSTK                    =00660090     82     (2)
DS$_LOGIC                      =00660070     82     (2)
DS$_MCHK                       =00660088     82     (2)    #-394    (3)
DS$_MMOFF                      =00660058     82     (2)
DS$_NEEDUNIT                   =006600F8     82     (2)
DS$_NODE                       =00660128     82     (2)
DS$_NOPCS                      =00660110     82     (2)
DS$_NORMAL                     =00660001     82     (2)
DS$_NOSUPPORT                  =006600B1     82     (2)
DS$_NOTDON                     =00660030     82     (2)
DS$_NOTIMP                     =006600B0     82     (2)      82     (2)
DS$_NULLSTR                    =00660010     82     (2)
DS$_OVERFLOW                   =00660008     82     (2)
DS$_POWER                      =00660098     82     (2)
DS$_PROGERR                    =00660020     82     (2)
DS$_SEVERE                     =00660004     82     (2)
DS$_TRANSL                     =006600A0     82     (2)
DS$_TRUNCATE                   =00660028     82     (2)
DS$_UNEXPINT                   =006600D8     82     (2)
DS$_VASFULL                    =00660048     82     (2)
DS$_WARNING                    =00660000     82     (2)      82     (2)
DSA$GL_PASSNO                   0000FE54                   #-601    (8)     #-975    (13)
ENV$M_DOMAIN                   =00000002     68     (2)
ENV$M_LEVEL                    =00000001     68     (2)
ENV$M_SUPER                    =000003FC     68     (2)
ENV$S_DOMAIN                   =00000001     68     (2)
ENV$S_LEVEL                    =00000001     68     (2)
ENV$S_SUPER                    =00000008     68     (2)
ENV$V_DOMAIN                   =00000001     68     (2)      68     (2)
ENV$V_LEVEL                    =00000000     68     (2)      68     (2)
ENV$V_SUPER                    =00000002     68     (2)
ENV$_CPU                       =00000000     68     (2)      68     (2)
ENV$_FUNCTIONAL                =00000000     68     (2)      68     (2)
ENV$_REPAIR                    =00000001     68     (2)      68     (2)
```

```
ENV$_SUPER                     =00000001     68      (2)
ENV$_SYSTEM                    =00000001     68      (2)     68     (2)
ERR$_MSGADR                    =0000000C     85      (2)
ERR$_NARGS                     =0000000A     85      (2)
ERR$_NUM                       =00000004     85      (2)
ERR$_P1                        =00000014     85      (2)   #-525    (7)    #-548    (7)    #-564    (7)    #-573    (7)
                                                           #-576    (7)
ERR$_P2                        =00000018     85      (2)     522    (7)    #-565    (7)
ERR$_P3                        =0000001C     85      (2)   #-523    (7)    #-525    (7)    #-566    (7)
ERR$_P4                        =00000020     85      (2)
ERR$_P5                        =00000024     85      (2)
ERR$_P6                        =00000028     85      (2)
ERR$_POINTER                   =00000010     85      (2)
ERR$_UNIT                      =00000008     85      (2)
F_ROWERR1                       00000742-R   486     (5)     536    (7)      554    (7)
F_ROWERR2                       0000075D-R   489     (5)     538    (7)      556    (7)
F_WCS_ERR                       0000085D-R   507     (5)     576    (7)
HALT                           =00000004     89      (2)
IPR                            =00000006     89      (2)
KA$W_WCS                        00000036                     417    (3)    #-599    (8)    #-973    (13)
KA_PTABLE                       00000000-XR                #-416    (3)    #-598    (8)    #-972    (13)
LOAD_MICRO                      0000038C-R   338     (3)     1023   (14)     1077   (15)
L_COUNT                         00000000-R   96      (2)   #-1020   (14)   #-1073   (15)   #-229    (3)    #-231    (3)
                                                           #-240    (3)    #-243    (3)    #-350    (3)    #-369    (3)
L_FLAGS                         00000010-R   101     (2)   #-1019   (14)   #-1072   (15)   #-381    (3)      577    (7)
                                                           #-670    (9)      687    (9)    #-689    (9)    #-692    (9)
                                                           #-738    (10)   #-757    (10)     759    (10)   #-804    (11)
                                                           #-821    (11)     823    (11)   #-897    (12)     904    (12)
                                                             925    (12)     938    (12)   #-939    (12)
L_INC                           00000004-R   97      (2)   #-317    (3)    #-900    (12)   #-941    (12)     942    (12)
L_PERM_FLAGS                    00000014-R   114     (2)     237    (3)      339    (3)    #-415    (3)      422    (3)
                                                           #-613    (8)    #-983    (13)
L_REV_CHECK                     00000000-XR                #-413    (3)
L_ROW                           00000008-R   98      (2)   #-320    (3)    #-898    (12)   #-945    (12)   #-947    (12)
MANUAL                         =00000001     89      (2)
MCHK                           =00000005     89      (2)
MICRO_DATA                      00000224-R   192     (2)     343    (3)
MICRO_WORD_COUNT                00000030-R   140     (2)   #-1073   (15)   #-439    (3)
MI_ERR                          0000098E-R   562     (7)     916    (12)     923    (12)
MSG_006                         00000002-R   585     (8)     597    (8)
MSG_007                         00000002-R   962     (13)    971    (13)
M_BIT19                        =00080000     121     (2)   #-358    (3)
M_BIT20                        =00100000     122     (2)   #-373    (3)    #-616    (8)
M_DATA_ERR                     =00000002     106     (2)   #-821    (11)
M_OLD_WCS                      =00000001     117     (2)   #-415    (3)
M_PAT                          =00000001     104     (2)   #-689    (9)    #-692    (9)
M_REVERSE                      =00000008     110     (2)   #-939    (12)
M_ROW_ERR                      =00000004     108     (2)   #-757    (10)
M_XFC_ERR                      =00000010     112     (2)   #-381    (3)
NOISY_WORDS                     00000144-R   162     (2)   #-263    (3)    #-291    (3)      810    (11)
PAR$M_ATDEF                    =00000010     83      (2)
PAR$M_ATHI                     =00000008     83      (2)
PAR$M_ATLO                     =00000004     83      (2)
PAR$M_NODEF                    =00000002     83      (2)
PAR$V_ATDEF                    =00000004     83      (2)
```

```
PAR$V_ATHI                        =00000003     83      (2)
PAR$V_ATLO                        =00000002     83      (2)
PAR$V_NODEF                       =00000001     83      (2)
PAR$_BIN                          =00000002     83      (2)
PAR$_DEC                          =0000000A     83      (2)
PAR$_HEX                          =00000010     83      (2)
PAR$_NO                           =00000000     83      (2)
PAR$_OCT                          =00000008     83      (2)
PAR$_YES                          =00000001     83      (2)
PCS                               =00000008     89      (2)
POWER_FAIL                        =00000003     89      (2)
PR$_SID                            00000000-XR                 #-411     (3)
PR750$_CADR                       =00000025                    #-606     (8)      #-954     (12)
PSL$K_EXEC                        =00000001     84      (2)
PSL$K_KERNEL                      =00000000     84      (2)
PSL$K_SUPER                       =00000002     84      (2)
PSL$K_USER                        =00000003     84      (2)
PSL$M_CBIT                        =00000001     84      (2)
PSL$M_NBIT                        =00000008     84      (2)
PSL$M_VBIT                        =00000002     84      (2)
PSL$M_ZBIT                        =00000004     84      (2)
PSL$V_CBIT                        =00000000     84      (2)
PSL$V_NBIT                        =00000003     84      (2)
PSL$V_VBIT                        =00000001     84      (2)
PSL$V_ZBIT                        =00000002     84      (2)
RENTRY1                            0000038D-R   951     (12)     895      (12)
RENTRY2                            000000CD-R   1031    (14)     1021     (14)
RENTRY3                            00000155-R   1085    (15)     1075     (15)
RM_DECODE                          0000028F-R   251     (3)      816      (11)
ROWERR                             000008E8-R   528     (7)      760      (10)
SCB$L_MACHCHK                      00000004                    #-1093    (15)     #-896     (12)     #-955     (12)     #-980     (13)
SCB$L_OPCCUS                       00000014                    #-1094    (15)     #-981     (13)
SEP_FUNCTIONAL                    =00000002     68      (2)
SEP_REPAIR                        =00000003     68      (2)
SETUP_ADDR                         00000350-R   316     (3)      901      (12)
SIZ...                            =00000001     84      (2)      68       (2)       83       (2)       84       (2)
SIZE_WCS                           0000043D-R   410     (3)      614      (8)       984      (13)
T6_S1                              00000094-R   669     (9)
T6_S1_X                            00000120-R   694     (9)
T6_S2                              0000012B-R   737     (10)
T6_S2_X                            000001B2-R   762     (10)
T6_S3                              000001BD-R   803     (11)
T6_S3_X                            00000248-R   827     (11)
T6_S4                              00000253-R   894     (12)
T6_S4_X                            000003B9-R   956     (12)
T7_S1                              0000007A-R   1018    (14)
T7_S1_X                            000000EE-R   1036    (14)
T7_S2                              000000F9-R   1071    (15)
T7_S2_X                            00000176-R   1091    (15)
TAG1                               0000027A-R   897     (12)     953      (12)
TAG2                               000003A9-R   954     (12)    #-950     (12)
TAG3                               0000009C-R   1022    (14)     1033     (14)
TAG4                               000000E8-R   1034    (14)    #-1028    (14)     #-1030    (14)
TAG5                               00000120-R   1076    (15)     1087     (15)
TAG6                               00000170-R   1088    (15)    #-1082    (15)     #-1084    (15)
```

```
TEST_006                          00000018-R     597     (8)       597     (8)
TEST_006_X                        000003C7-R     958     (12)    #-431     (3)      #-433    (3)      #-604    (8)
TEST_007                          0000001C-R     971     (13)      971     (13)
TEST_007_X                        00000196-R     1095    (15)    #-978     (13)
TEST_VALUE                        00000034-R     144     (2)     #-1076    (15)     #-440    (3)
TU58                             =00000002       89      (2)
T_ADR_ERR                         00000772-R     492     (5)       824     (11)
T_BAD_ATTACH                      00000568-R     460     (5)       430     (3)
T_BAD_ATTACH2                     000005DB-R     464     (5)       432     (3)
T_CS_PARITY1                      000004D5-R     451     (5)       952     (12)
T_CS_PARITY2                      00000508-R     454     (5)       1032    (14)
T_CS_PARITY3                      00000534-R     457     (5)       1086    (15)
T_DATAPATH                        00000691-R     474     (5)       683     (9)
T_DPERR                           000006B7-R     477     (5)       525     (7)        568    (7)
T_ENTRY_ERR                       0000080A-R     501     (5)       1029    (14)
T_MI_ERR1                         000007A0-R     495     (5)       916     (12)
T_MI_ERR2                         000007D5-R     498     (5)       923     (12)
T_NO_RESPONCE                     00000634-R     468     (5)       610     (8)
T_NO_WCS                          00000665-R     471     (5)       603     (8)        977    (13)
T_NXFIELD_ERR                     0000082F-R     504     (5)       1083    (15)
T_ROW                             000006F0-R     480     (5)       760     (10)
T_ROWERR                          00000717-R     483     (5)       530     (7)        546    (7)
T_XFC_ERR                         00000891-R     510     (5)       578     (7)
V_BIT14                          =0000000E       119     (2)
V_BIT15                          =0000000F       120     (2)     #-434     (3)
V_DATA_ERR                       =00000001       105     (2)     #-823     (11)
V_OLD_WCS                        =00000000       116     (2)     #-237     (3)      #-339    (3)      #-422    (3)
V_OVERFLOW_BIT                    00000020-R     128     (2)     #-323     (3)      #-434    (3)      #-815    (11)     #-942    (12)
V_PAT                            =00000000       103     (2)     #-687     (9)
V_REVERSE                        =00000003       109     (2)     #-904     (12)     #-925    (12)     #-938    (12)
V_ROW_ERR                        =00000002       107     (2)     #-759     (10)
V_XFC_ERR                        =00000004       111     (2)     #-577     (7)
WCS                              =00000007       89      (2)       597     (8)        971    (13)
WCS_ERR                           000009B4-R     571     (7)       1029    (14)       1083   (15)
WCS_MAX_SIZE_LWORDS               0000002C-R     137     (2)     #-243     (3)      #-437    (3)
WCS_MAX_SIZE_MWORDS               00000028-R     134     (2)     #-435     (3)      #-905    (12)     #-930    (12)
XFC_SERVICE                       00000410-R     380     (3)       981     (13)
```

```
                                             +------------------------+
                                             ! Macros Cross Reference !
                                             +------------------------+

MACRO             SIZE          DEFINITION       REFERENCES...
-----             ----          ----------       -------------
$D1_ABPROGRAM     1             396    (3)       396    (3)
$D1_BSUBT         1             669    (9)       1018   (14)      1071   (15)      669    (9)       737    (10)      803    (11)
                                                 894    (12)
$D1_BTEST         1             597    (8)       597    (8)       971    (13)
$D1_ERR           1             610    (8)       1029   (14)      1032   (14)      1083   (15)      1086   (15)      610    (8)
                                                 683    (9)       760    (10)      824    (11)      916    (12)      923    (12)
                                                 952    (12)
$D1_ERR_S         1             610    (8)       1029   (14)      1032   (14)      1083   (15)      1086   (15)      610    (8)
                                                 683    (9)       760    (10)      824    (11)      916    (12)      923    (12)
                                                 952    (12)
$D1_ESUBT         1             694    (9)       1036   (14)      1091   (15)      694    (9)       762    (10)      827    (11)
                                                 956    (12)
$D1_ETEST         1             958    (12)      1095   (15)      958    (12)
$D1_EXTEST        1             431    (3)       431    (3)       433    (3)       604    (8)       978    (13)
$D1_PRINT_S       1             430    (3)       430    (3)       432    (3)       525    (7)       530    (7)       536    (7)
                                                 538    (7)       546    (7)       554    (7)       556    (7)       568    (7)
                                                 576    (7)       578    (7)       603    (8)       977    (13)
$D1_SBTTL         2             585    (8)       1042   (15)      585    (8)       620    (9)       698    (10)      766    (11)
                                                 831    (12)      962    (13)      990    (14)
$D2_BDATA         1             597    (8)       597    (8)       971    (13)
$D2_BTEST         1             597    (8)       597    (8)       971    (13)
$D2_SBTTL         1             585    (8)       585    (8)       962    (13)
$DEF              1             84     (2)
$DEFINI           1             80     (2)       80     (2)       81     (2)       84     (2)       86     (2)       87     (2)
                                                 88     (2)       90     (2)
$DS_ABORT         1             396    (3)       396    (3)
$DS_BGNMESSAGE    1             521    (7)       521    (7)       529    (7)       545    (7)       563    (7)       572    (7)
$DS_BGNMOD        1             68     (2)       68     (2)
$DS_BGNSUB        1             669    (9)       1018   (14)      1071   (15)      669    (9)       737    (10)      803    (11)
                                                 894    (12)
$DS_BGNTEST       1             597    (8)       597    (8)       971    (13)
$DS_BREAK         1             924    (12)      1095   (15)      924    (12)      958    (12)
$DS_CKLOOP        1             611    (8)       1033   (14)      1087   (15)      611    (8)       684    (9)       953    (12)
$DS_CLRVEC_S      1             955    (12)      1093   (15)      1094   (15)      955    (12)
$DS_DSADEF        5             81     (2)       81     (2)
$DS_DSDEF         2             82     (2)       82     (2)
$DS_DSSDEF        8             86     (2)       86     (2)
$DS_ENDDATA       1             597    (8)       597    (8)       971    (13)
$DS_ENDMESSAGE    1             526    (7)       526    (7)       542    (7)       560    (7)       569    (7)       580    (7)
$DS_ENDMOD        1             1097   (15)      1097   (15)
$DS ENDSUB        1             694    (9)       1036   (14)      1091   (15)      694    (9)       762    (10)      827    (11)
                                                 956    (12)
$DS_ENDTEST       1             958    (12)      1095   (15)      958    (12)
$DS_ENVDEF        2             68     (2)       68     (2)
$DS_ERRDEF        1             85     (2)       85     (2)
$DS_ERRHARD_S     1             610    (8)       1029   (14)      1032   (14)      1083   (15)      1086   (15)      610    (8)
                                                 682    (9)       760    (10)      824    (11)      915    (12)      922    (12)
                                                 952    (12)
```

```
$DS_EXIT          1             431    (3)       431    (3)       433    (3)       604    (8)       978    (13)
$DS_HPODEF        2             87     (2)       87     (2)
$DS_KA_DEF        3             88     (2)       88     (2)
$DS_PAGE          1             583    (7)       1038   (14)      583    (7)       618    (8)       696    (9)       764    (10)
                                                 829    (11)      960    (12)      986    (13)
$DS_PARDEF        1             83     (2)       83     (2)
$DS_PRINTB_S      1             524    (7)       524    (7)       530    (7)       536    (7)       546    (7)       554    (7)
                                                 568    (7)       576    (7)       578    (7)
$DS_PRINTF_S      1             430    (3)       430    (3)       432    (3)       603    (8)       977    (13)
$DS_PRINTSIG_G    1             395    (3)       395    (3)
$DS_PRINTX_S      1             538    (7)       538    (7)       556    (7)
$DS_PROBE_S       1             608    (8)       608    (8)
$DS_PSLDEF        1             84     (2)       84     (2)
$DS_SBTTL         1             585    (8)       1042   (15)      585    (8)       620    (9)       698    (10)      766    (11)
                                                 831    (12)      962    (13)      990    (14)
$DS_SCBDEF        3             80     (2)       80     (2)
$DS_SECDEF        1             89     (2)       89     (2)
$DS_SETVEC_S      1             896    (12)      896    (12)      980    (13)      981    (13)
$EQU              1             84     (2)       68     (2)       82     (2)       83     (2)       84     (2)
$EQULS1           1             84     (2)       68     (2)       82     (2)       84     (2)
$EQULST           1             68     (2)       68     (2)       82     (2)       84     (2)
$GBLINI           2             68     (2)       68     (2)       82     (2)       83     (2)       84     (2)
$OFFDEF           1             85     (2)       85     (2)
$PR750DEF         1             90     (2)       90     (2)
$PSLDEF           2             84     (2)       84     (2)
$PUSHADR          1             610    (8)       1029   (14)      1032   (14)      1083   (15)      1086   (15)      610    (8)
                                                 683    (9)       760    (10)      824    (11)      916    (12)      923    (12)
                                                 952    (12)
$VIELD            1             68     (2)       68     (2)       83     (2)       84     (2)
$VIELD1           1             84     (2)       68     (2)       83     (2)       84     (2)
ECKAX_SECDEF      1             89     (2)       89     (2)

                                             +------------------------+
                                             ! Performance indicators !
                                             +------------------------+

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                    30    00:00:00.07    00:00:00.54
Command processing               897    00:00:00.69    00:00:01.23
Pass 1                          1317    00:00:12.68    00:00:23.43
Symbol table sort                  0    00:00:00.57    00:00:01.09
Pass 2                           603    00:00:03.19    00:00:07.32
Symbol table output               54    00:00:00.22    00:00:00.48
Psect synopsis output              5    00:00:00.03    00:00:00.03
Cross-reference output            40    00:00:01.08    00:00:02.57
Assembler run totals            2949    00:00:18.53    00:00:36.69

The working set limit was 2612 pages.
173596 bytes (340 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 548 non-local and 77 local symbols.
1098 source lines were read in Pass 1, producing 33 object records in Pass 2.
130 pages of virtual memory were used to define 57 macros.
```

```
                                              +---------------------------+
                                              ! Macro library statistics !
                                              +---------------------------+

Macro library name                             Macros defined
------------------                             --------------
DISK$DIAGPACK02:[TAYLOR.ECKAX]ECKAX.MLB;1                   1
SYS$COMMON:[SYSLIB]DIAG.MLB;955                            47
SYS$COMMON:[SYSLIB]STARLET.MLB;2                            9
TOTALS (all libraries)                                     57

897 GETS were required to define 57 macros.

There were no errors, warnings or information messages.

MACRO/LIS=ECKAX5.LIS/CROSS ECKAX5
```

Table of contents

```
    (1)        1        ECKAX6: MACHINE CHECK EXCEPTIONS AND INTERRUPTS
    (3)       61        DECLARATIONS
    (4)      158        ERROR MESSAGES
    (5)      252        DATA TABLES
    (8)      333        ERROR REPORTING ROUTINE
    (9)      424        SUBROUTINES
    (10)     645        TEST 8:  Machine Check Exceptions and Interrupts
    (11)     659        SUBTEST 8.1:  Bus Error ABORT
    (12)     736        SUBTEST 8.2:  Write Bus Error INTERRUPT
    (13)     813        SUBTEST 8.3:  WCS Parity Error ABORT
```

```
0000            .SBTTL  ECKAX6: MACHINE CHECK EXCEPTIONS AND INTERRUPTS
0000            .TITLE  ECKAX6  VAX-11/750 Specific CPU Cluster Exerciser
0000            .IDENT  /5.2/
0000
0000    ;
0000    ; Copyright (C) 1980,1983,1986
0000    ; Digital Equipment Corporation, Maynard, Massachusetts 01754
0000    ;
0000    ; This software is furnished  under a license for use only on a single
0000    ; computer  system and  may be  copied only with  the inclusion of the
0000    ; above copyright notice.  This software, or any other copies thereof,
0000    ; may not be provided or  otherwise made available to any other person
0000    ; except for use on such system and to one who agrees to these license
0000    ; terms.  Title to and  ownership of the  software  shall at all times
0000    ; remain in DEC.
0000    ;
0000    ; The information in this software is subject to change without notice
0000    ; and should  not be construed  as a commitment  by Digital  Equipment
0000    ; Corporation.
0000    ;
0000    ; DEC assumes  no  responsibility  for  the use or  reliability of its
0000    ; software on equipment which is not supplied by DEC.
0000    ;
0000
0000    ;++
0000    ; Facility:     VAX Architecture Diagnostic.
0000    ;
0000    ; Abstract:     This module tests most of the possible ways in which a machine
0000    ;               check exception or interrupt may occure.  If WCS is available,
0000    ;               it is used to force some of the machine checks.
0000    ;
0000    ; Environment:  VAX Diagnostic Supervisor.
0000    ;
0000    ; Author:       Fred Roemer     19-Feb-81       Version 2.2
0000    ;
0000    ;       NO CHANGES TO THIS MODULE FOR VERSION 3.3 (DOCUMENT CHANGE ONLY)
0000    ;
0000    ;
0000    ;       Dan Milleville  19-Mar-82               Version 3.4
0000    ;       Changed Test 3 exit code to accomodate temporarily a hardware problem.
0000    ;
0000    ;       Dave Shull      30-Nov-82               Version 4.0
0000    ;       Added tests 13 and 14 for coverage of PCS module.
0000    ;
0000    ;       Tai-Chun Pan    04-Mar-83               Version 5.0
0000    ;       Added test 5 for coverage of Processor Halts Architecture Undefined.
0000    ;       Added tests 11 and 12 for coverage of new ROM with MRSP of TU58.
0000    ;
0000    ;       Wayne R. Taylor February 1986           Version 5.2
0000    ;       Redefined IPR's to jive with starlet.
0000    ;       Modified the code for problem report Eckax5 where:
0000    ;               Tests 6 & 13 tests with cache off.
0000    ;               Subtest 1 does not check bit 0 of the memory under test.
0000    ;               Subtest 4 tests row 0, 1, 2, but not row 3.
0000    ;       Modifications were supplied by the submitter.
```

```
     0000     56 ;        Search on 5.2.
     0000     57 ;
     0000     58 ;--
```

```
               0000     60
               0000     61          .SBTTL  DECLARATIONS
               0000     62          .LIST   MEB
               0000     63          .NLIST  CND
           00000000     64          .PSECT  ECKAXF, PAGE, NOWRT
               0000     65          .LIBRARY         SYS$LIBRARY:DIAG ; VAX family diagnostic library.
               0000     66          $DS_BGNMOD       CEP_REPAIR, TN=8
               0000        :::      Macro-32 Diagnostic macro library Version 8.3  DIAG.MLB (245)
               0000     67
               0000     68 ;
               0000     69 ; Include files:
               0000     70 ;
               0000     71
               0000     72          .LIBRARY         ECKAX           ; CPU Cluster Exerciser library.
               0000     73
               0000     74
               0000     78
               0000     79          $DS_SCBDEF
               0000     80          $DS_DSDEF
               0000     81          $DS_DSADEF
               0000     82          $DS_DSSDEF
               0000     83          $DS_ERRDEF
               0000     84          $DS_PARDEF
               0000     85          $DS_PSLDEF
               0000     86          $DS_MPODEF
               0000     87          $DS_KA_DEF
               0000     88          ECKAX_SECDEF
               0000     89          $PR750DEF        ;5.2
               0000     90
               0000     91 ;
               0000     92 ; Own storage:
               0000     93 ;
               0000     94
               0000     95
00000004       0000     96 L_COUNT:        .BLKL                    ; used as a counter
00000008       0004     97 L_ADDRESS:      .BLKL                    ; used to temporary store addresses
0000000C       0008     98 L_CUR_ISP:      .BLKL                    ; used to store initial ISP
00000010       000C     99 RENTRY_POINT:   .BLKL                    ; used to store rentry address
00000014       0010    100 L_EXP_MOD:      .BLKL                    ; temp storage of expected mode
00000018       0014    101 L_EXP_STACK:    .BLKL                    ; temp storage of expected stack
0000001C       0018    102                 .BLKL
00000020       001C    103 L_TIMER:        .BLKL                    ; used as a timer for interrupts
               0020    104
00000024       0020    105 L_FLAGS:        .BLKL                    ; contains the following flags
               0024    106                                          ; bit positions and masks
00000000       0024    107 V_READY=0                                ; ready to execute flag
00000001       0024    108 M_READY=1@0                              ;
00000001       0024    109 V_DONE=1                                 ; restart done flag
00000002       0024    110 M_DONE=1@1                               ;
00000002       0024    111 V_EXECUTE_ERR=2                          ; execution error flag
00000004       0024    112 M_EXECUTE_ERR=1@2                        ;
00000003       0024    113 V_TIMER=3                                ; timer interrupt flag
00000008       0024    114 M_TIMER=1@3                              ;
00000004       0024    115 V_FATAL_ERR=4                            ; flag fatal error
00000010       0024    116 M_FATAL_ERR=1@4                          ;
```

```
00000005  0024    117 V_PSL_ERR=5                                        ; flag PSL error
00000020  0024    118 M_PSL_ERR=1@5                                      ;
00000006  0024    119 V_GPR_ERR=6                                        ; flag GPR error
00000040  0024    120 M_GPR_ERR=1@6                                      ;
00000007  0024    121 V_ERROR_FLAG=7                                     ; flag any error
00000080  0024    122 M_ERROR_FLAG=1@7                                   ;
00000008  0024    123 V_KSP_ERR=8                                        ; flag KSP error
00000100  0024    124 M_KSP_ERR=1@8                                      ;
00000009  0024    125 V_ISP_ERR=9                                        ; flag ISP error
00000200  0024    126 M_ISP_ERR=1@9                                      ;
0000000A  0024    127 V_SP_ERR=10                                        ; flag SP error
00000400  0024    128 M_SP_ERR=1@10                                      ;
0000000B  0024    129 V_ISTK=11                                          ; flag IS
00000800  0024    130 M_ISTK=1@11                                        ;
0000000C  0024    131 V_MC_ERR=12                                        ; flag machine check error
00001000  0024    132 M_MC_ERR=1@12                                      ;
0000000D  0024    133 V_INT=13                                           ; flag interrupt
00002000  0024    134 M_INT=1@13                                         ;
0000000E  0024    135 V_MCINT=14                                         ; flag machine check interrupt
00004000  0024    136 M_MCINT=1@14                                       ;
0000000F  0024    137 V_XFC_ERR=15                                       ; flag unexpected XFC exception
00008000  0024    138 M_XFC_ERR=1@15
          0024    139
00080000  0024    140 M_BIT19=1@19
00100000  0024    141 M_BIT20=1@20
00000080  0024    142 BUFSIZ=128                                         ; buffer size
00000060  0024    143 SCB$L_WTBUSERR=^X60                                ; write bus error interrupt vector
FFFFFFFC  0024    144 A_LAST_ADDRESS=-4                                  ; address for nonexistent memory
00000000  0024    145 L_DATA=0                                           ; data used to write nonexistent mem.
          0024    146
          0024    147
00F00000  0024    148 A_WCS_ADDRESS:  .LONG   ^XF00000                   ; starting address of wcs
          0028    149
00F0C000  0028    150 A_WCS_ENA_ADDR: .LONG   ^XF0C000                   ; address of WCS enable register
          002C    151
          002C    152
          002C    156
```

```
                                      002C     158 .SBTTL  ERROR MESSAGES
                                      002C     159 ;----------------------------------------------------------------
                                      002C     160 T_MC_MESS:
20 64 65 74 70 6D 65 74 74 41 20 00'  002C     161          .ASCIC    Attempted to cause a Machine Check by!/
4D 20 61 20 65 73 75 61 63 20 6F 74   0038
6B 63 65 68 43 20 65 6E 69 68 63 61   0044
                     2F 21 79 62 20   0050
                                 28   002C
                                      0055     162 ;----------------------------------------------------------------
                                      0055     163 T_BUSERR:
20 61 20 67 6E 69 64 61 65 72 20 00'  0055     164          .ASCIC    ` reading a nonexistent memory location!/!/`
20 74 6E 65 74 73 69 78 65 6E 6F 6E   0061
74 61 63 6F 6C 20 79 72 6F 6D 65 6D   006D
               2F 21 2F 21 6E 6F 69   0079
                                 2A   0055
                                      0080     165 ;----------------------------------------------------------------
                                      0080     166 T_WTBUSERR:
6F 74 20 67 6E 69 74 69 72 77 20 00'  0080     167          .ASCIC    ` writing to a nonexistent memory locaion!/!/
65 74 73 69 78 65 6E 6F 6E 20 61 20   008C
6F 6C 20 79 72 6F 6D 65 6D 20 74 6E   0098
         2F 21 2F 21 6E 6F 69 61 63   00A4
                                 2C   0080
                                      00AD     168 ;----------------------------------------------------------------
                                      00AD     169 T_WCS_ERR:
20 67 6E 69 74 75 63 65 78 65 20 00'  00AD     170          .ASCIC    executing out of WCS with bad parity!/!/
77 20 53 43 57 20 66 6F 20 74 75 6F   00B9
69 72 61 70 20 64 61 62 20 68 74 69   00C5
                  2F 21 2F 21 79 74   00D1
                                 29   00AD
                                      00D7     171 ;----------------------------------------------------------------
                                      00D7     172 T_MC_ERR:
6D 20 6E 69 20 72 6F 72 72 45 20 00'  00D7     173          .ASCIC    Error in machine check logout stack!/`-
6B 63 65 68 63 20 65 6E 69 68 63 61   00E3
63 61 74 73 20 74 75 6F 67 6F 6C 20   00EF
21 3A 4C 41 55 54 43 41 20 2F 21 6B   00FB
                                 2F   0107
                                 30   00D7
                                      0108     174                    ACTUAL:!/`
                                      0108     175 ;----------------------------------------------------------------
                                      0108     176 T_EXPECTED:
21 3A 44 45 54 43 45 50 58 45 20 00'  0108     177          .ASCIC    EXPECTED:!/`
                                 2F   0114
                                 0C   0108
                                      0115     178 ;----------------------------------------------------------------
                                      0115     179 T_UNEXP_DATA:
64 65 74 63 65 70 78 65 6E 55 20 00'  0115     180          .ASCIC    Unexpected data in machine check logout information
63 61 6D 20 6E 69 20 61 74 61 64 20   0121
6C 20 6B 63 65 68 63 20 65 6E 69 68   012D
6D 72 6F 66 6E 69 20 74 75 6F 67 6F   0139
                     6E 6F 69 74 61   0145
                                 34   0115
                                      014A     181 ;----------------------------------------------------------------
                                      014A     182 T_NOT_READY:
72 74 73 6E 69 20 74 73 65 54 20 00'  014A     183          .ASCIC    Test instruction was not ready to be executed.'/
6E 20 73 61 77 20 6E 6F 69 74 63 75   0156
```

```
20 6F 74 20 79 64 61 65 72 20 74 6F   0162
2E 64 65 74 75 63 65 78 65 20 65 62   016E
                               2F 21   017A
                                  31   014A
                                       017C    184 ;-------------------------------------------------------------------------------
                                       017C    185 T_EXECUTE:
72 74 73 6E 69 20 74 73 65 54 20 00'  017C    186          .ASCIC   Test instruction execution error.!/
75 63 65 78 65 20 6E 6F 69 74 63 75   0188
21 2E 72 6F 72 72 65 20 6E 6F 69 74   0194
                                  2F   01A0
                                  24   017C
                                       01A1    187 ;-------------------------------------------------------------------------------
                                       01A1    188 T_TIMER:
64 65 74 63 65 70 78 65 6E 55 20 00'  01A1    189          .ASCIC   Unexpected interval timer interrupt.!/
69 74 20 6C 61 76 72 65 74 6E 69 20   01AD
70 75 72 72 65 74 6E 69 20 72 65 6D   01B9
                         2F 21 2E 74   01C5
                                  27   01A1
                                       01C9    190 ;-------------------------------------------------------------------------------
                                       01C9    191 T_XFC:
64 65 74 63 65 70 78 65 6E 55 20 00'  01C9    192          .ASCIC   Unexpected XFC exception.!/
69 74 70 65 63 78 65 20 43 46 58 20   01D5
                      2F 21 2E 6E 6F   01E1
                                  1C   01C9
                                       01E6    193 ;-------------------------------------------------------------------------------
                                       01E6    194 T_KSP_ERR1:
20 50 53 20 6C 65 6E 72 65 4B 20 00'  01E6    195          .ASCIC   Kernel SP is incorrect - EXP = !XL  ACT = !XL!/
74 63 65 72 72 6F 63 6E 69 20 73 69   01F2
4C 58 21 20 3D 20 50 58 45 20 2D 20   01FE
21 4C 58 21 20 3D 20 54 43 41 20 20   020A
                                  2F   0216
                                  30   01E6
                                       0217    196 ;-------------------------------------------------------------------------------
                                       0217    197 T_KSP_ERR2:
20 50 53 20 6C 65 6E 72 65 4B 20 00'  0217    198          .ASCIC   Kernel SP - EXP =(UNPREDICTABLE) ACT = !XL!/
52 50 4E 55 28 3D 20 50 58 45 20 2D   0223
41 20 29 45 4C 42 41 54 43 49 44 45   022F
   2F 21 4C 58 21 20 3D 20 54 43       023B
                                  2D   0217
                                       0245    199 ;-------------------------------------------------------------------------------
                                       0245    200 T_ISP_ERR1:
20 74 70 75 72 72 65 74 6E 49 20 00'  0245    201          .ASCIC   Interrupt SP is incorrect - EXP = !XL  ACT = !XL!/
72 72 6F 63 6E 69 20 73 69 20 50 53   0251
20 3D 20 50 58 45 20 2D 20 74 63 65   025D
21 20 3D 20 54 43 41 20 20 4C 58 21   0269
                         2F 21 4C 58   0275
                                  33   0245
                                       0279    202 ;-------------------------------------------------------------------------------
                                       0279    203 T_ISP_ERR2:
20 74 70 75 72 72 65 74 6E 49 20 00'  0279    204          .ASCIC   Interrupt SP - EXP =(UNPREDICTABLE) ACT = !XL!/
55 28 3D 20 50 58 45 20 2D 20 50 53   0285
45 4C 42 41 54 43 49 44 45 52 50 4E   0291
21 4C 58 21 20 3D 20 54 43 41 20 29   029D
                                  2F   02A9
                                  30   0279
```

```
                                                 02AA    205 ;-------------------------------------------------------------------------
                                                 02AA    206 T_SP_ERR1:
6F 63 6E 69 20 73 69 20 50 53 20 00'  02AA    207          .ASCIC    SP is incorrect - EXP = !XL  ACT = !XL!/
20 50 58 45 20 2D 20 74 63 65 72 72   02B6
3D 20 54 43 41 20 20 4C 58 21 20 3D   02C2
                  2F 21 4C 58 21 20   02CE
                                  29  02AA
                                      02D4    208 ;-------------------------------------------------------------------------
                                      02D4    209 T_SP_ERR2:
2F 21 4C 58 21 20 3D 20 50 53 20 00'  02D4    210          .ASCIC    SP = !XL!/
                                  0B  02D4
                                      02E0    211 ;-------------------------------------------------------------------------
                                      02E0    212 T_GPR_ERR:
69 20 61 74 61 64 20 64 61 42 20 00'  02E0    213          .ASCIC    Bad data in general purpose register:!/
75 70 20 6C 61 72 65 6E 65 67 20 6E   02EC
74 73 69 67 65 72 20 65 73 6F 70 72   02F8
                     2F 21 3A 72 65   0304
                                  28  02E0
                                      0309    214 ;-------------------------------------------------------------------------
                                      0309    215 T_BAD_DATA:
58 21 20 3D 20 50 58 45 20 20 20 00'  0309    216          .ASCIC      EXP = !XL    ACT = !XL    XOR= !XL !/
21 20 3D 20 54 43 41 20 20 20 20 4C   0315
21 20 3D 52 4F 58 20 20 20 20 4C 58   0321
                     2F 21 20 4C 58   032D
                                  28  0309
                                      0332    217 ;-------------------------------------------------------------------------
                                      0332    218 T_GOOD_DATA:
58 21 20 3D 20 50 58 45 20 20 20 00'  0332    219          .ASCIC      EXP = !XL    ACT = !XL !/
21 20 3D 20 54 43 41 20 20 20 20 4C   033E
                     2F 21 20 4C 58   034A
                                  1C  0332
                                      034F    220 ;-------------------------------------------------------------------------
                                      034F    221 T_NOT_ON_KS:
72 65 4B 20 6E 6F 20 74 6F 4E 20 00'  034F    222          .ASCIC    Not on Kernel Stack as expected.!/
73 61 20 6B 63 61 74 53 20 6C 65 6E   035B
2F 21 2E 64 65 74 63 65 70 78 65 20   0367
                                  23  034F
                                      0373    223 ;-------------------------------------------------------------------------
                                      0373    224 T_NOT_ON_IS:
74 6E 49 20 6E 6F 20 74 6F 4E 20 00'  0373    225          .ASCIC    Not on Interrupt Stack as expected.!/
6B 63 61 74 53 20 74 70 75 72 72 65   037F
64 65 74 63 65 70 78 65 20 73 61 20   038B
                           2F 21 2E   0397
                                  26  0373
                                      039A    226 ;-------------------------------------------------------------------------
                                      039A    227 T_EXP_PSL:
50 20 64 65 74 63 65 70 78 45 20 00'  039A    228          .ASCIC    Expected PSL = !_!XL(X) ; !AC!/
58 28 4C 58 21 5F 21 20 3D 20 4C 53   03A6
      2F 21 43 41 21 20 3B 20 29      03B2
                                  20  039A
                                      03BB    229 ;-------------------------------------------------------------------------
                                      03BB    230 T_ACT_PSL:
4C 53 50 20 6C 61 75 74 63 41 20 00'  03BB    231          .ASCIC    Actual PSL = !_!XL(X) ; !AC!/
20 29 58 28 4C 58 21 5F 21 20 3D 20   03C7
               2F 21 43 41 21 20 3B   03D3
```

```
                                      1E  03BB
                                          03DA   232 ;-------------------------------------------------------------------------
                                          03DA   233 T_PSL:
58 21 5F 21 20 3D 20 4C 53 50 20 00'  03DA   234          .ASCIC    PSL = !_!XL (X) ; !AC!/
21 43 41 21 20 3B 20 29 58 28 20 4C   03E6
                                      2F  03F2
                                      18  03DA
                                          03F3   235 ;-------------------------------------------------------------------------
                                          03F3   236 T_GPR:
20 20 72 65 74 73 69 67 65 52 20 00'  03F3   237          .ASCIC   ' Register  Expected    Actual     XOR!/
20 20 20 20 64 65 74 63 65 70 78 45   03FF
58 20 20 20 20 20 6C 61 75 74 63 41   040B
                           2F 21 52 4F  0417
                                      27  03F3
                                          041B   238 ;-------------------------------------------------------------------------
                                          041B   239 T_XOR:
21 20 20 5F 21 43 41 21 20 20 20 00'  041B   240          .ASCIC   '    !AC!_  !XL   !XL   !XL!/
21 20 20 20 4C 58 21 20 20 20 4C 58   0427
                           2F 21 4C 58  0433
                                      1B  041B
                                          0437   241 ;-------------------------------------------------------------------------
                                          0437   242 T_NOXOR:
21 20 20 5F 21 43 41 21 20 20 20 00'  0437   243          .ASCIC        !AC!_  !XL   !XL!/'
      2F 21 4C 58 21 20 20 20 4C 58   0443
                                      15  0437
                                          044D   244 ;--------------------------------------------------------------------- -----
                                          044D   245 T_NO_WCS:
57 20 67 6E 69 74 69 78 45 2F 21 00'  044D   246          .ASCIC   !/Exiting WCS parity subtest, WCS last address = 0.!/
75 73 20 79 74 69 72 61 70 20 53 43   0459
6C 20 53 43 57 20 2C 74 73 65 74 62   0465
20 73 73 65 72 64 64 61 20 74 73 61   0471
                  2F 21 2E 30 20 3D   047D
                                      35  044D
                                          0483   247 ;-------------------------------------------------------------------------
                                          0483   248
                                          0483   249
                                          0483   250
```

```
                          0483    252 .SBTTL  DATA TABLES
                          0483    253
                          0483    254 ; Data table for test 1
                          0483    255
                00000000  0483    256 DATA_1:          .LONG    ^X0              ; expected PSL subtest 1
                          0487    257
                001D0000  0487    258 DATA_2:          .LONG    ^X1D0000         ; expected PSL subtest 2
                          048B    259
```

```
          048B
          048B  ; This microcode will write R5 into the D register and then decrement the D
          048B  ; register and write it back into R5. Since the second micro word has bad
          048B  ; parity, the operation should be unsucessful and a machine check should
          048B  ; occur when attempting to execute this out of WCS.
          048B
          048B  MICRO_DATA:
          048B                                                  ; first micro word
          048B                                                  ; R[R5]_WB,D_D-1,NEXT/2
00003FC2  048B                      .LONG   ^X03FC2
00054047  048F                      .LONG   ^X54047
00071203  0493                      .LONG   ^X71203
0004840E  0497                      .LONG   ^X4840E             ; This has bad parity (good = 8840E)
          049B
          049B                                                  ; second micro word
          049B                                                  ; ROT/ZERO,ALPCTL/WX_S,WCTRL/FLAGS_WB,NEXT -
cont> /2002
00001FC2  049B                      .LONG   ^X01FC2
00000307  049F                      .LONG   ^X00307
000B7003  04A3                      .LONG   ^XB7003
00088005  04A7                      .LONG   ^X88005
          04AB
          04AB                                                  ; third micro word
          04AB                                                  ; FLUSH XB,NEXT/2003
00001FC3  04AB                      .LONG   ^X01FC3
00000487  04AF                      .LONG   ^X00487
00092403  04B3                      .LONG   ^X92403
000481A5  04B7                      .LONG   ^X481A5
          04BB                                                  ; fourth micro word
          04BB                                                  ; D_R[R5],NEXT/2000
00001FC0  04BB                      .LONG   ^X01FC0
00054047  04BF                      .LONG   ^X54047
000BE603  04C3                      .LONG   ^XBE603
000C8005  04C7                      .LONG   ^XC8005
          04CB
          04CB
          04CB
0000000E  04CB  TABLE3: .LONG   14                  ; length for printsig
00660088  04CF          .LONG   DS$_MCHK            ; type of exception
00000028  04D3          .LONG   ^X28                ; length of logout stack
00000002  04D7          .LONG   ^X2                 ; error code
FFFFFFFC  04DB          .LONG   ^XFFFFFFFC          ; VA register
000000BB' 04DF          .LONG   TAG_3+7             ; PC at time of error
00000000  04E3          .LONG   ^X0                 ; MDR
0000000C  04E7          .LONG   ^XC                 ; saved mode rergister
00000000  04EB          .LONG   ^X0                 ; read lock timeout
00000000  04EF          .LONG   ^X0                 ; TB group parity error reg
00000001  04F3          .LONG   ^X1                 ; CACHE error register
00000008  04F7          .LONG   ^X8                 ; bus error register
00000008  04FB          .LONG   ^X8                 ; machine check summary reg
000000B4' 04FF          .LONG   TAG_3               ; interrupt / exception PC
00000000  0503          .LONG   ^X0                 ; interrupt / exception PSL
          0507
          0507
0000000E  0507  TABLE4: .LONG   14                  ; length for printsig
00660088  050B          .LONG   DS$_MCHK            ; type of exception
```

```
00000028  050F   316          .LONG    ^X28              ; length of logout stack
00000001  0513   317          .LONG    ^X1               ; error code
00000000  0517   318          .LONG    ^X0               ; VA register
00000291' 051B   319          .LONG    TAG_4+2           ; PC at time of error
00000AFA' 051F   320          .LONG    XFC_SERVICE+2     ; MDR
00000004  0523   321          .LONG    ^X4               ; saved mode rergister
00000000  0527   322          .LONG    ^X0               ; read lock timeout
00000000  052B   323          .LONG    ^X0               ; TB group parity error reg
00000000  052F   324          .LONG    ^X0               ; CACHE error register
00000000  0533   325          .LONG    ^X0               ; bus error register
00000000  0537   326          .LONG    ^X0               ; machine check summary reg
0000028F' 053B   327          .LONG    TAG_4             ; interrupt / exception PC
00000008  053F   328          .LONG    ^X8               ; interrupt / exception PSL
          0543   329
          0543   330
          0543   331
```

```
                              0543   333 .SBTTL ERROR REPORTING ROUTINE
                              0543   334
                              0543   335 ERR_MESSAGE:
                              0543   336         $DS_BGNMESSAGE  REGMASK=<R2,R7> ; Save R2 and R7
                         0084 0543                         .WORD   ^M<R2,R7>               ; ENTRY MASK
        5E   00000084 8F   C2 0545   337         SUBL2   #BUFSIZ+4,SP            ; Reserve room on stack for cvtreg
                 52   5E   D0 054C   338         MOVL    SP,R2                   ; Save starting address of buffer
                              054F   339         $DS_PRINTB_S    FORMAT=T_MC_MESS
             FAD9 CF   9F     054F                       PUSHAB  T_MC_MESS
        000100E0 9F   01   FB 0553                       CALLS   #$$N, @#DS$PRINTB
                              055A   340         $DS_PRINTB_S    FORMAT=@ERR$_P1(AP)
                14 BC   9F    055A                       PUSHAB  @ERR$_P1(AP)
        000100E0 9F   01   FB 055D                       CALLS   #$$N, @#DS$PRINTB
         32 FAB7 CF   0C   E1 0564   341         BBC     #V_MC_ERR,L_FLAGS,1$     ; br = no machine check error
                              056A   342         $DS_PRINTB_S    FORMAT=T_MC_ERR
             FB69 CF   9F     056A                       PUSHAB  T_MC_ERR
        000100E0 9F   01   FB 056E                       CALLS   #$$N, @#DS$PRINTB
   57   38   00000000'EF   C1 0575   343         ADDL3   A_MCHK_BUFFER,#56,R7     ; load address of actual data
        67     FF4A CF   7D   057D   344         MOVQ    TABLE3,(R7)              ; load size and type
                              0582   345         $DS_PRINTSIG_G  ARGPTR=(R7)
        00010100 9F   67   FA 0582                       CALLG   (R7), @#DS$PRINTSIG
                              0589   346         $DS_PRINTB_S    FORMAT=T_EXPECTED
             FB7B CF   9F     0589                       PUSHAB  T_EXPECTED
        000100E0 9F   01   FB 058D                       CALLS   #$$N, @#DS$PRINTB
                              0594   347         $DS_PRINTSIG_G  ARGPTR=@ERR$_P2(AP)
     00010100 9F   18 BC   FA 0594                       CALLG   @ERR$_P2(AP), @#DS$PRINTSIG
         0B FA7F CF   00   E0 059C   348 1$:     BBS     #V_READY,L_FLAGS,3$      ; br = instruction ready to execute
                              05A2   349         $DS_PRINTB_S    FORMAT=T_NOT_READY
             FBA4 CF   9F     05A2                       PUSHAB  T_NOT_READY
        000100E0 9F   01   FB 05A6                       CALLS   #$$N, @#DS$PRINTB
         0B FA6E CF   02   E1 05AD   350 3$:     BBC     #V_EXECUTE_ERR,L_FLAGS,4$ ; br = instruction executed
                              05B3   351         $DS_PRINTB_S    FORMAT=T_EXECUTE
             FBC5 CF   9F     05B3                       PUSHAB  T_EXECUTE
        000100E0 9F   01   FB 05B7                       CALLS   #$$N, @#DS$PRINTB
         1F FA5D CF   0B   E1 05BE   352 4$:     BBC     #V_ISTK,L_FLAGS,5$       ; br = on kernel stack
         2A FA57 CF   08   E1 05C4   353         BBC     #V_KSP_ERR,L_FLAGS,6$    ; br = no KSP error
                              05CA   354         $DS_PRINTB_S    FORMAT=T_KSP_ERR1,P0-L_EXP_KSP,P1=L_ACT_KSP
           00000000'EF   DD   05CA                       PUSHL   L_ACT_KSP
           00000000'EF   DD   05D0                       PUSHL   L_EXP_KSP
             FC0C CF   9F     05D6                       PUSHAB  T_KSP_ERR1
        000100E0 9F   03   FB 05DA                       CALLS   #$$N, @#DS$PRINTB
                      11   11 05E1   355         BRB     6$
                              05E3   356 5$:     $DS_PRINTX_S    FORMAT=T_KSP_ERR2,P0=L_ACT_KSP
           00000000'EF   DD   05E3                       PUSHL   L_ACT_KSP
             FC2A CF   9F     05E9                       PUSHAB  T_KSP_ERR2
        000100E8 9F   02   FB 05ED                       CALLS   #$$N, @#DS$PRINTX
         1F FA27 CF   0B   E0 05F4   357 6$:     BBS     #V_ISTK,L_FLAGS,7$       ; br = on interrupt stack
         2A FA21 CF   09   E1 05FA   358         BBC     #V_ISP_ERR,L_FLAGS,8$    ; br = no ISP error
                              0600   359         $DS_PRINTB_S    FORMAT=T_ISP_ERR1,P0=L_EXP_ISP,P1=L_ACT_ISP
           00000000'EF   DD   0600                       PUSHL   L_ACT_ISP
           00000000'EF   DD   0606                       PUSHL   L_EXP_ISP
             FC35 CF   9F     060C                       PUSHAB  T_ISP_ERR1
        000100E0 9F   03   FB 0610                       CALLS   #$$N, @#DS$PRINTB
                      11   11 0617   360         BRB     8$                       ; skip extened error report
                              0619   361 7$:     $DS_PRINTX_S    FORMAT=T_ISP_ERR2,P0=L_ACT_ISP
```

```
          00000000'EF    DD  0619                        PUSHL    L_ACT_ISP
              FC56 CF    9F  061F                        PUSHAB   T_ISP_ERR2
      000100E8 9F    02  FB  0623                        CALLS    #$$N, @#DS$PRINTX
       19 F9F1 CF    0A  E1  062A   362 8$:      BBC      #V_SP_ERR,L_FLAGS,9$     ; br = no SP error
                             0630   363          $DS_PRINTB_S     FORMAT=T_SP_ERR1,P0=L_EXP_SP,P1=L_ACT_SP
          00000000'EF    DD  0630                        PUSHL    L_ACT_SP
          00000000'EF    DD  0636                        PUSHL    L_EXP_SP
              FC6A CF    9F  063C                        PUSHAB   T_SP_ERR1
      000100E0 9F    03  FB  0640                        CALLS    #$$N, @#DS$PRINTB
                     11  11  0647   364          BRB      10$                      ; skip extended error report
                             0649   365 9$:      $DS_PRINTX_S     FORMAT=T_SP_ERR2,P0=L_ACT_SP
          00000000'EF    DD  0649                        PUSHL    L_ACT_SP
              FC81 CF    9F  064F                        PUSHAB   T_SP_ERR2
      000100E8 9F    02  FB  0653                        CALLS    #$$N, @#DS$PRINTX
       03 F9C1 CF    05  E0  065A   366 10$:     BBS      #V_PSL_ERR,L_FLAGS,11$   ; br = PSL error
                   008A  31  0660   367          BRW      12$                      ; skip messages
                             0663   368 11$:     $DS_CVTREG_S     MSB=#31,DATA=L_EXP_PSL,MNEADR=A_PSLMNE,-
                             0663   369                           STRBUF=(R2),MAXLEN=#BUFSIZ,-
                             0663   370                           V1=MODELIST,V2=MODELIST
                     00  DD  0663                        PUSHL    #0
                     00  DD  0665                        PUSHL    #0
                     00  DD  0667                        PUSHL    #0
                     00  DD  0669                        PUSHL    #0
          00000000'EF    DF  066B                        PUSHAL   MODELIST
          00000000'EF    DF  0671                        PUSHAL   MODELIST
          00000080 8F    DD  0677                        PUSHL    #BUFSIZ
                     62  9F  067D                        PUSHAB   (R2)
          00000000'EF    9F  067F                        PUSHAB   A_PSLMNE
          00000000'EF    DD  0685                        PUSHL    L_EXP_PSL
                     1F  DD  068B                        PUSHL    #31
      000100B0 9F    0B  FB  068D                        CALLS    #11, @#DS$CVTREG
                             0694   371          $DS_PRINTB_S     FORMAT=T_EXP_PSL,P0=L_EXP_PSL,P1=R2
                     52  DD  0694                        PUSHL    R2
          00000000'EF    DD  0696                        PUSHL    L_EXP_PSL
              FCFA CF    9F  069C                        PUSHAB   T_EXP_PSL
      000100E0 9F    03  FB  06A0                        CALLS    #$$N, @#DS$PRINTB
                             06A7   372          $DS_CVTREG_S     MSB=#31,DATA=L_ACT_PSL,MNEADR=A_PSLMNE,-
                             06A7   373                           STRBUF=(R2),MAXLEN=#BUFSIZ,-
                             06A7   374                           V1=MODELIST,V2=MODELIST
                     00  DD  06A7                        PUSHL    #0
                     00  DD  06A9                        PUSHL    #0
                     00  DD  06AB                        PUSHL    #0
                     00  DD  06AD                        PUSHL    #0
          00000000'EF    DF  06AF                        PUSHAL   MODELIST
          00000000'EF    DF  06B5                        PUSHAL   MODELIST
          00000080 8F    DD  06BB                        PUSHL    #BUFSIZ
                     62  9F  06C1                        PUSHAB   (R2)
          00000000'EF    9F  06C3                        PUSHAB   A_PSLMNE
          00000000'EF    DD  06C9                        PUSHL    L_ACT_PSL
                     1F  DD  06CF                        PUSHL    #31
      000100B0 9F    0B  FB  06D1                        CALLS    #11, @#DS$CVTREG
                             06D8   375          $DS_PRINTB_S     FORMAT=T_ACT_PSL,P0=L_ACT_PSL,P1=R2
                     52  DD  06D8                        PUSHL    R2
          00000000'EF    DD  06DA                        PUSHL    L_ACT_PSL
              FCD7 CF    9F  06E0                        PUSHAB   T_ACT_PSL
              000100E0 9F   03   FB  06E4                       CALLS   #$$N, @#DS$PRINTB
                            44   11  06EB   376         BRB     13$
                                     06ED   377 12$:    $DS_CVTREG_S    MSB=#31,DATA=L_EXP_PSL,MNEADR=A_PSLMNE,-
                                     06ED   378                         STRBUF=(R2),MAXLEN=#BUFSIZ,-
                                     06ED   379                         V1=MODELIST,V2=MODELIST
                            00   DD  06ED                       PUSHL   #0
                            00   DD  06EF                       PUSHL   #0
                            00   DD  06F1                       PUSHL   #0
                            00   DD  06F3                       PUSHL   #0
                   00000000'EF   DF  06F5                       PUSHAL  MODELIST
                   00000000'EF   DF  06FB                       PUSHAL  MODELIST
                   00000080 8F   DD  0701                       PUSHL   #BUFSIZ
                            62   9F  0707                       PUSHAB  (R2)
                   00000000'EF   9F  0709                       PUSHAB  A_PSLMNE
                   00000000'EF   DD  070F                       PUSHL   L_EXP_PSL
                            1F   DD  0715                       PUSHL   #31
              000100B0 9F   0B   FB  0717                       CALLS   #11, @#DS$CVTREG
                                     071E   380         $DS_PRINTX_S    FORMAT=T_PSL,P0=L_EXP_PSL,P1=R2
                            52   DD  071E                       PUSHL   R2
                   00000000'EF   DD  0720                       PUSHL   L_EXP_PSL
                       FCB0 CF   9F  0726                       PUSHAB  T_PSL
              000100E8 9F   03   FB  072A                       CALLS   #$$N, @#DS$PRINTX
               5C F8EA CF   06   E1  0731   381 13$:    BBC     #V_GPR_ERR,L_FLAGS,17$ ; br = no GPR error
                                     0737   382         $DS_PRINTB_S    FORMAT=T_GPR
                       FCB8 CF   9F  0737                       PUSHAB  T_GPR
              000100E0 9F   01   FB  073B                       CALLS   #$$N, @#DS$PRINTB
              53   00000000'EF   DE  0742   383         MOVAL   GPR_TABLE,R3            ; load address of expected data
              54   00000000'EF   D0  0749   384         MOVL    A_MCHK_BUFFER,R4        ; load address of actual data
                            56   7C  0750   385         CLRQ    R6                      ; init R5 and R6 for counter and data
                       57   0D   D0  0752   386         MOVL    #13,R7                  ; load limit
            58   00000000'EF45   DE  0755   387 14$:    MOVAL   REG_TABLE[R5],R8        ; load address of ascic string
              56   6445   6345   CD  075D   388         XORL3   (R3)[R5],(R4)[R5],R6    ; get XOR of data if bad
                            17   13  0763   389         BEQL    15$                     ; br = good data
                                     0765   390         $DS_PRINTB_S    FORMAT=T_XOR,P0=R8,P1=(R3)[R5],P2=(R4)[R5],P3=R6
                            56   DD  0765                       PUSHL   R6
                          6445   DD  0767                       PUSHL   (R4)[R5]
                          6345   DD  076A                       PUSHL   (R3)[R5]
                            58   DD  076D                       PUSHL   R8
                       FCA8 CF   9F  076F                       PUSHAB  T_XOR
              000100E0 9F   05   FB  0773                       CALLS   #$$N, @#DS$PRINTB
                            13   11  077A   391         BRB     16$                     ; check if finished
                                     077C   392 15$:    $DS_PRINTX_S    FORMAT=T_NOXOR,P0=R8,P1=(R3)[R5],P2=(R4)[R5]
                          6445   DD  077C                       PUSHL   (R4)[R5]
                          6345   DD  077F                       PUSHL   (R3)[R5]
                            58   DD  0782                       PUSHL   R8
                       FCAF CF   9F  0784                       PUSHAB  T_NOXOR
              000100E8 9F   04   FB  0788                       CALLS   #$$N, @#DS$PRINTX
                    C2 55   57   F3  078F   393 16$:    AOBLEQ  R7,R5,14$
               0B F888 CF   03   E1  0793   394 17$:    BBC     #V_TIMER,L_FLAGS,18$    ; br = no timer error
                                     0799   395         $DS_PRINTB_S    FORMAT=T_TIMER
                       FA04 CF   9F  0799                       PUSHAB  T_TIMER
              000100E0 9F   01   FB  079D                       CALLS   #$$N, @#DS$PRINTB
               0B F877 CF   0F   E1  07A4   396 18$:    BBC     #V_XFC_ERR,L_FLAGS,19$  ; br = no timer error
                                     07AA   397         $DS_PRINTB_S    FORMAT=T_XFC
                       FA1B CF   9F  07AA                       PUSHAB  T_XFC
```

```
   000100E0 9F    01    FB  07AE                        CALLS    #$$N, @#DS$PRINTB
                            07B5    398 19$:
                            07B5    399         $DS_ENDMESSAGE
                      04    07B5                        RET                       ; RETURN TO DIAGNOSTIC SUPERVISOR
                            07B6    400
                            07B6    401
                            07B6    402 ERR2_MESSAGE:
                            07B6    403         $DS_BGNMESSAGE  REGMASK=<R2,R7> ; Save R2 and R7
                    0084    07B6                        .WORD    ^M<R2,R7>                ; ENTRY MASK
                            07B8    404         $DS_PRINTB_S    FORMAT=T_MC_MESS
          F870 CF    9F     07B8                        PUSHAB   T_MC_MESS
   000100E0 9F    01    FB  07BC                        CALLS    #$$N, @#DS$PRINTB
                            07C3    405         $DS_PRINTB_S    FORMAT=@ERR$_P1(AP)
             14 BC   9F     07C3                        PUSHAB   @ERR$_P1(AP)
   000100E0 9F    01    FB  07C6                        CALLS    #$$N, @#DS$PRINTB
                            07CD    406         $DS_PRINTB_S    FORMAT=T_MC_ERR
          F906 CF    9F     07CD                        PUSHAB   T_MC_ERR
   000100E0 9F    01    FB  07D1                        CALLS    #$$N, @#DS$PRINTB
   57    00000000'EF    D0  07D8    407         MOVL    A_MCHK_BUFFER,R7          ; load address of actual data
       67    FCE8 CF    7D  07DF    408         MOVQ    TABLE3,(R7)               ; load size and type
                            07E4    409         $DS_PRINTSIG_G  ARGPTR=(R7)
   00010100 9F    67    FA  07E4                        CALLG    (R7), @#DS$PRINTSIG
                            07EB    410         $DS_PRINTB_S    FORMAT=T_EXPECTED
          F919 CF    9F     07EB                        PUSHAB   T_EXPECTED
   000100E0 9F    01    FB  07EF                        CALLS    #$$N, @#DS$PRINTB
                            07F6    411         $DS_PRINTSIG_G  ARGPTR=@ERR$_P2(AP)
00010100 9F    18 BC    FA  07F6                        CALLG    @ERR$_P2(AP), @#DS$PRINTSIG
    0B F81D CF    00    E0  07FE    412 1$:     BBS     #V_READY,L_FLAGS,3$       ; br = instruction ready to execute
                            0804    413         $DS_PRINTB_S    FORMAT=T_NOT_READY
          F942 CF    9F     0804                        PUSHAB   T_NOT_READY
   000100E0 9F    01    FB  0808                        CALLS    #$$N, @#DS$PRINTB
    0B F80C CF    03    E1  080F    414 3$:     BBC     #V_TIMER,L_FLAGS,4$       ; br = no timer error
                            0815    415         $DS_PRINTB_S    FORMAT=T_TIMER
          F988 CF    9F     0815                        PUSHAB   T_TIMER
   000100E0 9F    01    FB  0819                        CALLS    #$$N, @#DS$PRINTB
    0B F7FB CF    0F    E1  0820    416 4$:     BBC     #V_XFC_ERR,L_FLAGS,5$     ; br = no timer error
                            0826    417         $DS_PRINTB_S    FORMAT=T_XFC
          F99F CF    9F     0826                        PUSHAB   T_XFC
   000100E0 9F    01    FB  082A                        CALLS    #$$N, @#DS$PRINTB
                            0831    418 5$:
                            0831    419         $DS_ENDMESSAGE
                      04    0831                        RET                       ; RETURN TO DIAGNOSTIC SUPERVISOR
                            0832    420
                            0832    421
                            0832    422
```

```
                                        0832     424 .SBTTL  SUBROUTINES
                                        0832     425
                                        0832     426 ; This routine will load R0 thur R11 with data patterns and read the
                                        0832     427 ; AP and FP to check after causing a machine check.
                                        0832     428
                                        0832     429 LOAD_GPR:
            5B    00000000'EF     DE    0832     430         MOVAL   GPR_TABLE,R11    ; load address of data table to load GPR's
                        50   8B   7D    0839     431         MOVQ    (R11)+,R0        ; move data into R0 and R1
                        52   8B   7D    083C     432         MOVQ    (R11)+,R2        ; move data into R2 and R3
                        54   8B   7D    083F     433         MOVQ    (R11)+,R4        ; move data into R4 and R5
                        56   8B   7D    0842     434         MOVQ    (R11)+,R6        ; move data into R6 and R7
                        58   8B   7D    0845     435         MOVQ    (R11)+,R8        ; move data into R8 and R9
                        5A   8B   7D    0848     436         MOVQ    (R11)+,R10       ; move data into R10 and R11
            00000000'EF      5C   D0    084B     437         MOVL    AP,L_EXP_AP      ; move contents of AP into expected data table
            00000000'EF      5D   D0    0852     438         MOVL    FP,L_EXP_FP      ; move contents of FP into expected data table
                                  05    0859     439         RSB                      ; return
                                        085A     440
                                        085A     441 ; This routine will read general purpose registers R0 to R11 , the AP, FP, SP
                                        085A     442 ; KSP, ISP and PSL into a buffer area (A_MCHK_BUFFER) to check data patterns
                                        085A     443 ; and data after a machine check exception. It will then retrun to to address
                                        085A     444 ; contained in RENTRY_POINT.
                                        085A     445
                                        085A     446         .ALIGN  LONG
                                        085C     447 READ_GPR:
            00000000'FF      50   7D    085C     448         MOVQ    R0,@A_MCHK_BUFFER ; move contents of R0 and R1 into buffer
            50    00000000'EF     D0    0863     449         MOVL    A_MCHK_BUFFER,R0 ; load R0 with address of buffer
                        50   08   C0    086A     450         ADDL2   #8,RC            ; get over first two entries
                        80   52   7D    086D     451         MOVQ    R2,(R0)+         ; move contents of R2 and R3 into buffer
                        80   54   7D    0870     452         MOVQ    R4,(R0)+         ; move R4 and R5
                        80   56   7D    0873     453         MOVQ    R6,(R0)+         ; move R6 and R7
                        80   58   7D    0876     454         MOVQ    R8,(R0)+         ; move R8 and R9
                        80   5A   7D    0879     455         MOVQ    R10,(R0)+        ; move R10 and R11
                        80   5C   D0    087C     456         MOVL    AP,(R0)+         ; move AP
                        80   5D   D0    087F     457         MOVL    FP,(R0)+         ; move FP
            00000000'EF      5E   D0    0882     458         MOVL    SP,L_ACT_SP      ; move SP
00000000'EF       00000000'8F     DB    0889     459         MFPR    #PR$_KSP,L_ACT_KSP ; move KSP
00000000'EF       00000000'8F     DB    0894     460         MFPR    #PR$_ISP,L_ACT_ISP ; move ISP
                  00000000'EF     DC    089F     461         MOVPSL  L_ACT_PSL        ; move PSL
                        50   08   C0    08A5     462         ADDL2   #8,R0            ; save room for printsig size and type
                             57   D4    08A8     463         CLRL    R7               ; init counter
                        80   8E   D0    08AA     464 1$:     MOVL    (SP)+,(R0)+      ; move MC logout into buffer
                     F9 57   0C   F3    08AD     465         AOBLEQ  #12,R7,1$        ; br : not finished
                     F757 DF      17    08B1     466         JMP     @RENTRY_POINT    ; return to subtest
                                        08B5     467
                                        08B5     468
                                        08B5     469
                                        08B5     470 ; This routine is used to service a WCS parity error machine check. The machine
                                        08B5     471 ; check logout is read off the stack into A_MCHK_BUFFER and checked after
                                        08B5     472 ; returning.
                                        08B5     473
                                        08B5     474         .ALIGN  LONG
                                        08B8     475 READ_MCHK:
            50    00000000'EF     D0    08B8     476         MOVL    A_MCHK_BUFFER,R0 ; load R0 with address of buffer
                        50   08   C0    08BF     477         ADDL2   #8,R0            ; save room for printsig size and type
                             57   D4    08C2     478         CLRL    R7               ; init counter
```

```
               55     0C    D0  08C4    479             MOVL     #12,R5               ; set limit
               80     8E    D0  08C7    480 1$:         MOVL     (SP)+,(R0)+          ; move MC logout into buffer
            F9 57     55    F3  08CA    481             AOBLEQ   R5,R7,1$             ; br = not finished
               5E     08    C2  08CE    482             SUBL2    #8,SP                ; back up SP to REI
                      6E    D6  08D1    483             INCL     (SP)                 ; update return address
                            02  08D3    484             REI                           ; return to subtest
                                08D4    485
                                08D4    486
                                08D4    487 ; This routine will read general purpose registers R0 to R11 , the AP, FP, SP
                                08D4    488 ; KSP, ISP and PSL into a buffer area (A_MCHK_BUFFER) to check data patterns
                                08D4    489 ; and data after a machine check interrupt.
                                08D4    490
                                08D4    491             .ALIGN   LONG
                                08D4    492 WTBUS_SERVICE:
         00000000'FF      50    7D  08D4    493             MOVQ     R0,@A_MCHK_BUFFER ; move contents of R0 and R1 into buffer
         50    00000000'EF      D0  08DB    494             MOVL     A_MCHK_BUFFER,R0 ; load R0 with address of buffer
               50     08    C0  08E2    495             ADDL2    #8,R0                ; get over first two entries
               80     52    7D  08E5    496             MOVQ     R2,(R0)+             ; move contents of R2 and R3 into buffer
               80     54    7D  08E8    497             MOVQ     R4,(R0)+             ; move R4 and R5
               80     56    7D  08EB    498             MOVQ     R6,(R0)+             ; move R6 and R7
               80     58    7D  08EE    499             MOVQ     R8,(R0)+             ; move R8 and R9
               80     5A    7D  08F1    500             MOVQ     R10,(R0)+            ; move R10 and R11
               80     5C    D0  08F4    501             MOVL     AP,(R0)+             ; move AP
               80     5D    D0  08F7    502             MOVL     FP,(R0)+             ; move FP
         00000000'EF      5E    D0  08FA    503             MOVL     SP,L_ACT_SP          ; move SP
 00000000'EF    00000000'8F     DB  0901    504             MFPR     #PR$_KSP,L_ACT_KSP ; move KSP
 00000000'EF    00000000'8F     DB  090C    505             MFPR     #PR$_ISP,L_ACT_ISP ; move ISP
                00000000'EF     DC  0917    506             MOVPSL   L_ACT_PSL            ; move PSL
      F6FA CF   00004000 8F     C8  091D    507             BISL2    #M_MCINT,L_FLAGS ; set interrupt flag
                            02  0926    508 1$:         REI                           ; return
                                0927    509
                                0927    510
                                0927    511 ; This routine is called after execution of an instruction which should cause
                                0927    512 ; a machine check. The MACHINE CHECK LOGOUT, PSL, SP, KSP, ISP, GENERAL
                                0927    513 ; PURPOSE REGISTERS and FLAG BYTE are examined for expected information. If
                                0927    514 ; an error exists, a flag is set to indicate that it should be reported.
                                0927    515
                                0927    516
                                0927    517 ERROR_CHECK:
                      57    D4  0927    518             CLRL     R7                   ; initialize counter
               55     0D    D0  0929    519             MOVL     #13,R5               ; set limit
         53    00000000'EF      D0  092C    520             MOVL     A_MCHK_BUFFER,R3 ; load address of actual data to check
         54    00000000'EF      DE  0933    521             MOVAL    GPR_TABLE,R4         ; load address of expected data
               84     83    D1  093A    522 1$:         CMPL     (R3)+,(R4)+          ; check data
                      09    13  093D    523             BEQL     2$                   ; br = data is good
      F6D8 CF   000000C0 8F     C8  093F    524             BISL2    #M_ERROR_FLAG!M_GPR_ERR,L_FLAGS ; set GPR and error flag
            EE 57     55    F3  0948    525 2$:         AOBLEQ   R5,R7,1$             ; br = not finished
       24 F6CF CF     0D    E0  094C    526             BBS      #V_INT,L_FLAGS,5$ ; br = checking an interrupt
               53     08    C0  0952    527             ADDL2    #8,R3                ; get over size and type to check machine check
                      57    D4  0955    528             CLRL     R7                   ; init counter
               55     0C    D0  0957    529             MOVL     #12,R5               ; set limit
               86     83    D1  095A    530 3$:         CMPL     (R3)+,(R6)+          ; check MC logout data
                      13    13  095D    531             BEQL     4$                   ; br = data good
               08     57    D1  095F    532             CMPL     R7,#8                ; cache data ?
                      0E    13  0962    533             BEQL     4$                   ; ignore cache condition
```

```
                    02   57   D1  0964   534           CMPL     R7,#2               ; virtual address register?
                         09   13  0967   535           BEQL     4$                  ; ignore va condition
      F6AE CF   00001080 8F   C8  0969   536           BISL2    #M_ERROR_FLAG!M_MC_ERR,L_FLAGS ; set error flag
                   E4 57 55   F3  0972   537 4$:       AOBLEQ   R5,R7,3$            ; br = not finished
    00000000'EF 00000000'EF   D1  0976   538 5$:       CMPL     L_ACT_SP,L_EXP_SP ; check SP
                         09   13  0981   539           BEQL     6$                  ; br = SF is good
      F694 CF   00000480 8F   C8  0983   540           BISL2    #M_SP_ERR!M_ERROR_FLAG,L_FLAGS ; set SP and error flag
        1F 00000000'EF   1A   E1  098C   541 6$:       BBC      #PSL$V_IS,L_ACT_PSL,7$ ; br = on KS no possible error
      F683 CF   00000800 8F   C8  0994   542           BISL2    #M_ISTK,L_FLAGS ; set interrupt stack flag
    00000000'EF 00000000'EF   D1  099D   543           CMPL     L_ACT_KSP,L_EXP_KSP ; check KSP
                         09   13  09A8   544           BEQL     7$                  ; br = KSP is good
      F66D CF   00000180 8F   C8  09AA   545           BISL2    #M_KSP_ERR!M_ERROR_FLAG,L_FLAGS ; set KSP and error flag
             16 F668 CF  0B   E0  09B3   546 7$:       BBS      #V_ISTK,L_FLAGS,8$ ; br = on IS no possible error
    00000000'EF 00000000'EF   D1  09B9   547           CMPL     L_ACT_ISP,L_EXP_ISP ; check ISP
                         09   13  09C4   548           BEQL     8$                  ; br = ISP is good
      F651 CF   00000280 8F   C8  09C6   549           BISL2    #M_ISP_ERR!M_ERROR_FLAG,L_FLAGS ; set ISP and error flag
    00000000'EF 00000000'EF   D1  09CF   550 8$:       CMPL     L_ACT_PSL,L_EXP_PSL ; check PSL
                         3E   13  09DA   551           BEQL     11$                 ; br = PSL is good
                    C2   18   EF  09DC   552           EXTZV    #PSL$V_CURMOD,#PSL$S_CURMOD,-
      F629 CF   00000000'EF       09DF   553                    L_EXP_PSL,L_EXP_MOD ; extract current mode from expected psl
                    02   18   ED  09E7   554           CMPZV    #PSL$V_CURMOD,#PSL$S_CURMOD,-
      F61E CF   00000000'EF       09EA   555                    L_ACT_PSL,L_EXP_MOD ; check current mode for error
                         18   12  09F2   556           BNEQ     9$                  ; br = wrong mode
                    01   1A   EF  09F4   557           EXTZV    #PSL$V_IS,#1,-
      F615 CF   00000000'EF       09F7   558                    L_EXP_PSL,L_EXP_STACK ; extract stack from expected PSL
                    01   1A   ED  09FF   559           CMPZV    #PSL$V_IS,#1,-
      F60A CF   00000000'EF       0A02   560                    L_ACT_PSL,L_EXP_STACK ; check current stack for error
                         05   13  0A0A   561           BEQL     10$                 ; br = mode is good
                 F60F CF  10  C8  0A0C   562 9$:       BISL2    #M_FATAL_ERR,L_FLAGS ; signal fatal error
      F606 CF   000000A0 8F   C8  0A11   563 10$:      BISL2    #M_PSL_ERR!M_ERROR_FLAG,L_FLAGS ; set PSL and error flag
             09 F601 CF  00   E0  0A1A   564 11$:      BBS      #V_READY,L_FLAGS,12$ ; br = no ready error
      F5F7 CF   00000080 8F   C8  0A20   565           BISL2    #M_ERROR_FLAG,L_FLAGS ; set error flag
             09 F5F2 CF  02   E1  0A29   566 12$:      BBC      #V_EXECUTE_ERR,L_FLAGS,13$ ; br = no execute error
      F5E8 CF   00000080 8F   C8  0A2F   567           BISL2    #M_ERROR_FLAG,L_FLAGS ; set error flag
             09 F5E3 CF  03   E1  0A38   568 13$:      BBC      #V_TIMER,L_FLAGS,14$ ; br = no unexpected timer interrupt
      F5D9 CF   00000080 8F   C8  0A3E   569           BISL2    #M_ERROR_FLAG,L_FLAGS ; set error flag
                              05  0A47   570 14$:      RSB                          ; return
                                  0A48   571
                                  0A48   572
                                  0A48   573 ; This routine is used to check errors after causing a machine check from
                                  0A48   574 ; a WCS parity error.
                                  0A48   575
                                  0A48   576 ERROR_CHECK1:
         53     00000000'EF   D0  0A48   577           MOVL     A_MCHK_BUFFER,R3 ; load address of actual data to check
                    53   08   C0  0A4F   578           ADDL2    #8,R3               ; get over size and type
                    86   83   D1  0A52   579           CMPL     (R3)+,(R6)+         ; check for machine check logout
                         10   12  0A55   580           BNEQU    3$                  ; br = not a machine check logout
                    86   83   D1  0A57   581           CMPL     (R3)+,(R6)+         ; check error summary parameter
                         0B   12  0A5A   582           BNEQU    3$                  ; br = incorrect error summary parameter
                    53   24   C0  0A5C   583           ADDL2    #36,R3              ; Point to PC of Machine check
                    56   24   C0  0A5F   584           ADDL2    #36,R6              ; Point to Expected PC of Machine check
                    86   83   D1  0A62   585           CMPL     (R3)+,(R6)+         ; check MC logout data
                         09   13  0A65   586           BEQLU    4$                  ; br = correct PC
      F5B0 CF   00001080 8F   C8  0A67   587 3$:       BISL2    #M_ERROR_FLAG!M_MC_ERR,L_FLAGS ; set error flag
             09 F5AB CF  00   E0  0A70   588 4$:       BBS      #V_READY,L_FLAGS,5$ ; br = no ready error
```

```
     F5A1 CF    00000080 8F   C8  0A76   589          BISL2   #M_ERROR_FLAG,L_FLAGS ; set error flag
        09 F59C CF      0F   E1  0A7F   590 5$:       BBC     #V_XFC_ERR,L_FLAGS,6$   ; br = no XFC error
     F592 CF    00000080 8F   C8  0A85   591          BISL2   #M_ERROR_FLAG,L_FLAGS ; set error flag
        09 F58D CF      03   E1  0A8E   592 6$:       BBC     #V_TIMER,L_FLAGS,7$ ; br = no unexpected timer interrupt
     F583 CF    00000080 8F   C8  0A94   593          BISL2   #M_ERROR_FLAG,L_FLAGS ; set error flag
                              05  0A9D   594 7$:       RSB                          ; return
                                  0A9E   595
                                  0A9E   596
                                  0A9E   597 ; This routine will service any unexpected interval timer interrupts.
                                  0A9E   598
                                  0A9E   599          .ALIGN  LONG
                                  0AA0   600 TIMER_SERVICE:
                 00'      00   DA  0AA0   601          MTPR    #0,S^#PR$_ICCS             ; Turn off the timer
             F578 CF      08   C8  0AA3   602          BISL2   #M_TIMER,L_FLAGS           ; set error flag
                              02  0AA8   603          REI                                ; return
                                  0AA9   604
                                  0AA9   605
                                  0AA9   606 ; This routine will clear the buffer area.
                                  0AA9   607
                                  0AA9   608 CLEAR_BUFFER:
          55    00000000'FF   DE  0AA9   609          MOVAL   @A_MCHK_BUFFER,R5 ; get starting buffer address
                   F54C CF   D4  0AB0   610          CLRL    L_COUNT           ; initialize counter
                        85   D4  0AB4   611 1$:       CLRL    (R5)+             ; clear a buffer location
   F4 F541 CF    00000080 8F   F3  0AB6   612          AOBLEQ  #128,L_COUNT,1$ ; br = not finished
                              05  0AC0   613          RSB                     ; return
                                  0AC1   614
                                  0AC1   615
                                  0AC1   616 ; This subroutine will load WCS with executable micro code with bad parity.
                                  0AC1   617
                                  0AC1   618 LOAD_MICRO:
          55    00000000'8F   DB  0AC1   619          MFPR    #PR$_SID,R5       ; get the system id register
     56   55      08      08   EF  0AC8   620          EXTZV   #8,#8,R5,R6       ; extract the revision level
          56    00000000'EF   D1  0ACD   621          CMPL    L_REV_CHECK,R6    ; check the current microcode revision level
                        04   1B  0AD4   622          BLEQU   1$                ; branch if NOT L0008 module
                   F54E DF   D4  0AD6   623          CLRL    @A_WCS_ENA_ADDR           ; Enable writing of WCS
                        5A   D4  0ADA   624 1$:       CLRL    R10                       ; initialize a counter
             53    F544 CF   D0  0ADC   625          MOVL    A_WCS_ADDRESS,R3          ; load WCS starting address
             54    F9A6 CF   DE  0AE1   626          MOVAL   MICRO_DATA,R4             ; load address of micro code to be loaded
                   83    84   D0  0AE6   627 10$:      MOVL    (R4)+,(R3)+               ; load 20 bits of micro word
                F9 5A    10   F2  0AE9   628          AOBLSS  #16,R10,10$               ; br = until all words done
     F52E DF    00100000 8F   C8  0AED   629          BISL2   #M_BIT20,@A_WCS_ADDRESS ; enable WCS
                              05  0AF6   630          RSB                             ; return
                                  0AF7   631
                                  0AF7   632
                                  0AF7   633 ; This routine is used to service unxepected XFC exceptions.
                                  0AF7   634
                                  0AF7   635 .ALIGN  LONG
                                  0AF8   636
                                  0AF8   637 XFC_SERVICE:
     F51F CF    00008000 8F   C8  0AF8   638          BISL2   #M_XFC_ERR,L_FLAGS ; set error flag
                        6E   D6  0B01   639          INCL    (SP)               ; update return address
                              02  0B03   640          REI                        ; return
                                  0B04   641
                                  0B04   642
                                  0B04   643          $DS_PAGE
                                  0B04                    .SBTTL  TEST 8:  Machine Check Exceptions and Interrupts
                              00000000                            .PSECT  TEST_008, PAGE, NOWRT
                                  002C
                                  002C   646 ;++
                                  002C   647 ; Test description:
                                  002C   648 ;
                                  002C   649 ;            This test checks some ways a machine chech exception or
                                  002C   650 ;            interrupt may occure.
                                  002C   651 ;
                                  002C   652 ;--
                                  002C   653
                                  002C   654          $DS_BGNTEST       <MCHK,ALL,DEFAULT>
                                  002C       DATA_008:
                        00000000  002C                    .LONG   0                ; TEST ARGUMENT TABLE TERMINATOR
                                  0030       TEST_008::
                            0000  0030                    .WORD   ^M<>             ; ENTRY MASK
                                  0032   655          $DS_SETVEC_S      VECTOR=#SCB$L_TIMER, SRVADR=TIMER_SERVICE
                      00      DD  0032                    PUSHL   #0
             00000AA0'EF      DF  0034                    PUSHAL  TIMER_SERVICE
             000000C0 8F      DD  003A                    PUSHL   #SCB$L_TIMER
     00010160 9F      03      FB  0040                    CALLS   #3, @#DS$SETVEC
                                  0047   656 1$:
                                  0047   657          $DS_PAGE
```

```
                                        0047              .SBTTL  SUBTEST 8.1:  Bus Error ABORT
                                        0047     660 ;+
                                        0047     661 ; Subtest description:
                                        0047     662 ;
                                        0047     663 ;        This subtest forces a bus error by attempting to read a nonexistent
                                        0047     664 ;        memory location, causing a machine check. The machine check logout
                                        0047     665 ;        pushed on the stack is checked for the correct information.
                                        0047     666 ;
                                        0047     667 ; Subtest steps:
                                        0047     668 ;
                                        0047     669 ;        SUBTEST BUS ERROR ABORT
                                        0047     670 ;        :  Get Machine Check vector to capture exception
                                        0047     671 ;        :  IF unsucessful
                                        0047     672 ;        :  :  THEN
                                        0047     673 ;        :  :  Report system error
                                        0047     674 ;        :  :  Abort test
                                        0047     675 ;        :  ENDIF
                                        0047     676 ;        :  Setup expected data and load GPR's with data
                                        0047     677 ;        :  Set ready to execute flag
                                        0047     678 ;        :  Access nonexistent memory (to cause exception)
                                        0047     679 ;        :  Set execution error flag
                                        0047     680 ;        :  IF no exception
                                        0047     681 ;        :  :  THEN
                                        0047     682 ;        :  :  Set execution error flag
                                        0047     683 ;        :  ENDIF
                                        0047     684 ;        :  Check machine check logout and register data
                                        0047     685 ;        :  IF error is detected
                                        0047     686 ;        :  :  THEN
                                        0047     687 ;        :  :  Report data error
                                        0047     688 ;        :  ENDIF
                                        0047     689 ;        :  IF fatal error flag is set
                                        0047     690 ;        :  :  THEN
                                        0047     691 ;        :  :  Abort test
                                        0047     692 ;        :  ENDIF
                                        0047     693 ;        :  Release Machine Check vector
                                        0047     694 ;        :  IF unsucessful
                                        0047     695 ;        :  :  THEN
                                        0047     696 ;        :  :  Report system error
                                        0047     697 ;        :  :  Abort test
                                        0047     698 ;        :  ENDIF
                                        0047     699 ;        :  Push return PC and PSL on stack
                                        0047     700 ;        :  Execute an REI
                                        0047     701 ;        ENDSUBTEST BUS ERROR ABORT
                                        0047     702 ;-
                                        0047     703
                                        0047     704          $DS_BGNSUB
                                        0047         T8_S1::
  00010030 9F    00000000'EF      FA    0047                         CALLG   $S$, @#DS$BGNSUB
                 00000AA9'EF      16    0052     705          JSB    CLEAR_BUFFER            ; clear buffer area
                       26    0F   DA    0058     706          MTPR   #^XF,#PR750$_MCESR      ; clear any previous machine checks 5.2
                                        005B     707          $DS_SETVEC_S    VECTOR=#SCB$L_MACHCHK, SRVADR=READ_GPR
                             00   DD    005B                         PUSHL   #0
                 0000085C'EF      DF    005D                         PUSHAL  READ_GPR
                             04   DD    0063                         PUSHL   #SCB$L_MACHCHK
         00010160 9F         03   FB    0065                         CALLS   #3, @#DS$SETVEC
```

```
     0000000C'EF    000000C2'8F   D0  006C    708 1$:      MOVL     #RENTRY3,RENTRY_POINT    ; load rentry address
                    00000020'EF   D4  0077    709          CLRL     L_FLAGS                  ; init flags
                    00000000'EF   D4  007D    710          CLRL     L_EXP_SP                 ; init expected SP
         00000000'EF    5E    34  C3  0083    711          SUBL3    #52,SP,L_EXP_SP          ; load expected SP
     00000000'EF    00000000'8F   DB  008B    712          MFPR     #PR$_ISP,L_EXP_ISP       ; load expected ISP
                    00000000'EF   D4  0096    713          CLRL     L_EXP_KSP                ; load expected KSP
     00000000'EF    00000483'EF   D0  009C    714          MOVL     DATA_1,L_EXP_PSL         ; load expected PSL
                    00000832'EF   16  00A7    715          JSB      LOAD_GPR                 ; load GPR's with data
               00000020'EF    01  C8  00AD    716          BISL2    #M_READY,L_FLAGS         ; set ready to execute flag
               55   FFFFFFFC'EF   D0  00B4    717 TAG_3:   MOVL     A_LAST_ADDRESS,R5        ; generate machine check
               00000020'EF    04  C8  00BB    718          BISL2    #M_EXECUTE_ERR,L_FLAGS   ; set execution error flag (should
                                      00C2    719                                            ;  skip over if exception occurs)
                                      00C2    720 RENTRY3:
                                      00C2    721          $DS_CLRVEC_S     VECTOR=#SCB$L_MACHCHK
                              04  DD  00C2                          PUSHL    #SCB$L_MACHCHK
               00010168 9F    01  FB  00C4                          CALLS    #1, @#DS$CLRVEC
                              56  D4  00CB    722 3$:      CLRL     R6                       ; init R6
         56   08    000004CB'8F   C1  00CD    723          ADDL3    #TABLE3,#8,R6            ; load R6 with address of exp data
                    00000927'EF   16  00D5    724          JSB      ERROR_CHECK              ; go check logout info for errors
           38 00000020'EF     07  E1  00DB    725          BBC      #V_ERROR_FLAG,L_FLAGS,5$ ; br = no error
                                      00E3    726          $DS_ERRHARD_S    UNIT=#0,PRLINK=ERR_MESSAGE,P1=#T_BUSERR,P2=#TABLE3
                    000004CB'8F   DD  00E3                          PUSHL    #TABLE3
                    00000055'8F   DD  00E9                          PUSHL    #T_BUSERR
                    00000543'EF   DF  00EF                          PUSHAL   ERR_MESSAGE
                              00  DD  00F5                          PUSHL    #0
                              00  DD  00F7                          PUSHL    #0
                              01  DD  00F9                          PUSHL    #$ER
                                      00FB        ;;; TEST 8, SUBTEST 1, ERROR 1
               000100D0 9F    06  FB  00FB                          CALLS    #$$M, @#DS$ERRHARD
                                      0102    727          $DS_BCOMPLETE    4$
                           07 50  E8  0102                          BLBS     R0, 4$
                                      0105    728          $DS_ABORT
               00010020 9F    6C  FA  0105                          CALLG    (AP), @#DS$ABORT
           07 00000020'EF     04  E1  010C    729 4$:      BBC      #V_FATAL_ERR,L_FLAGS,5$ ; br = no fatal errors
                                      0114    730          $DS_ABORT
               00010020 9F    6C  FA  0114                          CALLG    (AP), @#DS$ABORT
                                      011B    731 5$:
                                      011B    732          $DS_ENDSUB
                                      011B        T8_S1_X:
     00010038 9F    00000000'EF   FA  011B                          CALLG    $S$, @#DS$ENDSUB
                                      0126    733
                                      0126    734          $DS_PAGE
```

```
                                     0126                   .SBTTL  SUBTEST 8.2:  Write Bus Error INTERRUPT
                                     0126   737 ;+
                                     0126   738 ; Subtest description:
                                     0126   739 ;
                                     0126   740 ;       This subtest forces a write bus error by writing to a nonexistent
                                     0126   741 ;       memory location. An interrupt is generated and the interrupt pc
                                     0126   742 ;       is checked.
                                     0126   743 ;
                                     0126   744 ; Subtest steps:
                                     0126   745 ;
                                     0126   746 ;       SUBTEST WRITE BUS ERROR INTERRUPT
                                     0126   747 ;       :  Get Machine Check vector to capture exception
                                     0126   748 ;       :  IF unsucessful
                                     0126   749 ;       :  :  THEN
                                     0126   750 ;       :  :  Report system error
                                     0126   751 ;       :  :  Abort test
                                     0126   752 ;       :  ENDIF
                                     0126   753 ;       :  Setup expected data and load GPR's with data
                                     0126   754 ;       :  Set ready to execute flag
                                     0126   755 ;       :  Access nonexistent memory (to cause exception)
                                     0126   756 ;       :  Set execution error flag
                                     0126   757 ;       :  IF no exception
                                     0126   758 ;       :  :  THEN
                                     0126   759 ;       :  :  Set execution error flag
                                     0126   760 ;       :  ENDIF
                                     0126   761 ;       :  Check machine check logout and register data
                                     0126   762 ;       :  IF error is detected
                                     0126   763 ;       :  :  THEN
                                     0126   764 ;       :  :  Report data error
                                     0126   765 ;       :  ENDIF
                                     0126   766 ;       :  IF fatal error flag is set
                                     0126   767 ;       :  :  THEN
                                     0126   768 ;       :  :  Abort test
                                     0126   769 ;       :  ENDIF
                                     0126   770 ;       :  Release Machine Check vector
                                     0126   771 ;       :  IF unsucessful
                                     0126   772 ;       :  :  THEN
                                     0126   773 ;       :  :  Report system error
                                     0126   774 ;       :  :  Abort test
                                     0126   775 ;       :  ENDIF
                                     0126   776 ;       :  Push return PC and PSL on stack
                                     0126   777 ;       :  Execute an REI
                                     0126   778 ;       ENDSUBTEST WRITE BUS ERROR INTERRUPT
                                     0126   779 ;-
                                     0126   780
                                     0126   781         $DS_BGNSUB
                                     0126       T8_S2::
     00010030 9F   0000000C'EF   FA  0126                       CALLG   $S$, @#DS$BGNSUB
                   00000AA9'EF   16  0131   782         JSB     CLEAR_BUFFER            ; clear data buffer
                       26    0F  DA  0137   783         MTPR    #^XF,#PR750$_MCESR      ; clear any previous machine checks 5.2
                                     013A   784         $DS_SETVEC_S    VECTOR=#SCB$L_WTBUSERR, SRVADR=WTBUS_SERVICE
                             00  DD  013A                       PUSHL   #0
                   000008D4'EF   DF  013C                       PUSHAL  WTBUS_SERVICE
                   00000060 8F   DD  0142                       PUSHL   #SCB$L_WTBUSERR
              00010160 9F    03  FB  0148                       CALLS   #3, @#DS$SETVEC
```

```
                  00000020'EF   D4  014F    785 1$:     CLRL    L_FLAGS                    ; init flags
                  00000832'EF   16  0155    786         JSB     LOAD_GPR                   ; load GPR's with data
                  00000000'EF   D4  015B    787         CLRL    L_EXP_SP                   ; init expected SP
          00000000'EF   5E   08   C3  0161    788         SUBL3   #8,SP,L_EXP_SP             ; load expected SP
     00000000'EF   00000000'8F   DB  0169    789         MFPR    #PR$_ISP,L_EXP_ISP         ; load expected ISP
                  00000000'EF   D4  0174    790         CLRL    L_EXP_KSP                  ; clear KSP unpredictable
     00000000'EF   00000487'EF   D0  017A    791         MOVL    DATA_2,L_EXP_PSL           ; load expected PSL
     0000001C'EF   00002710 8F   D0  0185    792         MOVL    #10000,L_TIMER             ; set timer
              00000020'EF    01   C8  0190    793         BISL2   #M_READY,L_FLAGS           ; set ready to execute flag
     FFFFFFFC'EF   00000000'EF   D0  0197    794         MOVL    L_DATA,A_LAST_ADDRESS      ; generate machine check interrupt
          0E 00000020'EF    0E   E0  01A2    795 2$:     BBS     #V_MCINT,L_FLAGS,3$        ; br = interrupt has been serviced
               F1 0000001C'EF   F4  01AA    796         SOBGEQ  L_TIMER,2$                 ; br = no time out yet
              00000020'EF    04   C8  01B1    797         BISL2   #M_EXECUTE_ERR,L_FLAGS     ; set execution error flag (should not
                                      01B8    798                                            ;  get here if exception occurs)
                                      01B8    799 3$:     $DS_CLRVEC_S    VECTOR=#SCB$L_WTBUSERR
                  00000060 8F   DD  01B8                        PUSHL   #SCB$L_WTBUSERR
              00010168 9F    01   FB  01BE                        CALLS   #1, @#DS$CLRVEC
     00000020'EF   00002000 8F   C8  01C5    800 4$:     BISL2   #M_INT,L_FLAGS             ; set interrupt flag
                  00000927'EF   16  01D0    801         JSB     ERROR_CHECK                ; go check logout info for errors
          32 00000020'EF    07   E1  01D6    802         BBC     #V_ERROR_FLAG,L_FLAGS,6$ ; br = no error
                                      01DE    803         $DS_ERRHARD_S   UNIT=#0,PRLINK=ERR_MESSAGE,P1=#T_WTBUSERR
                  00000080'8F   DD  01DE                        PUSHL   #T_WTBUSERR
                  00000543'EF   DF  01E4                        PUSHAL  ERR_MESSAGE
                           00   DD  01EA                        PUSHL   #0
                           00   DD  01EC                        PUSHL   #0
                           01   DD  01EE                        PUSHL   #$ER
                                      01F0        ;;; TEST 8, SUBTEST 2, ERROR 1
              000100D0 9F    05   FB  01F0                        CALLS   #$$M, @#DS$ERRHARD
                                      01F7    804         $DS_BCOMPLETE   5$
                        07 50   E8  01F7                        BLBS    R0, 5$
                                      01FA    805         $DS_ABORT
              00010020 9F    6C   FA  01FA                        CALLG   (AP), @#DS$ABORT
          07 00000020'EF    04   E1  0201    806 5$:     BBC     #V_FATAL_ERR,L_FLAGS,6$ ; br = no fatal errors
                                      0209    807         $DS_ABORT
              00010020 9F    6C   FA  0209                        CALLG   (AP), @#DS$ABORT
                                      0210    808 6$:
                                      0210    809         $DS_ENDSUB
                                      0210        T8_S2_X:
     00010038 9F   0000000C'EF   FA  0210                        CALLG   $S$, @#DS$ENDSUB
                                      021B    810         $DS_PAGE
                                      021B    811
```

```
                                021B             .SBTTL  SUBTEST 8.3:  WCS Parity Error ABORT
                                021B   814 ;+
                                021B   815 ; Subtest description:
                                021B   816 ;
                                021B   817 ;         This subtest checks that WCS loaded with a micro word that has bad
                                021B   818 ;         parity will cause a machine check. First a set of micro words with bad
                                021B   819 ;         parity are written into the WCS. Next execution of the micro word is
                                021B   820 ;         attempted, which should cause a control store parity error machine
                                021B   821 ;         check. The machine check logout is checked to contain the correct
                                021B   822 ;         information.
                                021B   823 ;
                                021B   824 ; Subtest steps:
                                021B   825 ;
                                021B   826 ;         SUBTEST WCS Parity Error ABORT
                                021B   827 ;         :  Initialize flags
                                021B   828 ;         :  Write 3 micro words, one with bad parity
                                021B   829 ;         :  Use XFC instruction to execute out of WCS
                                021B   830 ;         :  Capture the machine check
                                021B   831 ;         :  IF machine check logout is incorrect
                                021B   832 ;         :  :  THEN
                                021B   833 ;         :  :  Report error
                                021B   834 ;         :  ENDIF
                                021B   835 ;         ENDSUBTEST WCS Parity Error ABORT
                                021B   836 ;
                                021B   837 ;
                                021B   838 ;--
                                021B   839
                                021B   840           $DS_BGNSUB
                                021B       T8_S3::
     00010030 9F   00000018'EF   FA   021B                  CALLG   $S$, @#DS$BGNSUB
                                0226   841
           51   00000000'EF   D0   0226   842           MOVL    KA_PTABLE,R1     ; load address of p table
                       36 A1   B5   022D   843           TSTW    KA$W_WCS(R1)     ; check to see if WCS is there
                          19   12   0230   844           BNEQ    2$               ; br = yes, test WCS
           0000FE54'EF    01   D1   0232   845           CMPL    #1,DSA$GL_PASSNO ; check for first pass
                          0D   12   0239   846           BNEQ    1$               ; br = not first pass
                                023B   847           $DS_PRINTF_S    FORMAT=T_NO_WCS
                0000044D'EF   9F   023B                  PUSHAB  T_NO_WCS
           000100F0 9F    01   FB   0241                  CALLS   #$$N, @#DS$PRINTF
                                0248   848 1$:       $DS_EXIT         SUB
                        00A5   31   0248                  BRW     T8_S3_X ; EXIT SUBTEST 3
                                024B   849
                00000020'EF   D4   024B   850 2$:       CLRL    L_FLAGS            ; initialize flags
                00000AC1'EF   16   0251   851           JSB     LOAD_MICRO         ; load wcs with micro code
                00000AA9'EF   16   0257   852           JSB     CLEAR_BUFFER       ; clear buffer area
                                025D   853           $DS_SETVEC_S    VECTOR=#SCB$L_OPCCUS,SRVADR=XFC_SERVICE,CODE=#2
                          02   DD   025D                  PUSHL   #2
                00000AF8'EF   DF   025F                  PUSHAL  XFC_SERVICE
                          14   DD   0265                  PUSHL   #SCB$L_OPCCUS
           00010160 9F    03   FB   0267                  CALLS   #3, @#DS$SETVEC
                    26    0F   DA   026E   854           MTPR    #^XF,#PR750$_MCESR      ; clear any previous machine checks 5.2
                                0271   855           $DS_SETVEC_S    VECTOR=#SCB$L_MACHCHK, SRVADR=READ_MCHK,CODE=#1
                          01   DD   0271                  PUSHL   #1
                000008B8'EF   DF   0273                  PUSHAL  READ_MCHK
                          04   DD   0279                  PUSHL   #SCB$L_MACHCHK
```

```
              00010160 9F  03    FB  027B                       CALLS   #3, @#DS$SETVEC
                   00000020'EF   D4  0282   856         CLRL    L_FLAGS                 ; init flags
              00000020'EF  01    C8  0288   857         BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
                                 FC  028F   858 TAG_4:  XFC                     ; generate machine check
                                 01  0290   859         NOP                     ; skip a location
                                 01  0291   860         NOP                     ; skip a location
                                 01  0292   861         NOP                     ; skip a location
                                 01  0293   862         NOP                     ; skip a location
              57   00000024'EF   D0  0294   863         MOVL    A_WCS_ADDRESS,R7; load address of first WCS location in R7
                            67   D4  029B   864         CLRL    (R7)            ; Clear the first location to insure that
                                     029D   865                                 ; WCS is turned off.
                                     029D   866         $DS_CLRVEC_S    VECTOR=#SCB$L_MACHCHK
                            04   DD  029D                       PUSHL   #SCB$L_MACHCHK
              00010168 9F   01   FB  029F                       CALLS   #1, @#DS$CLRVEC
                                     02A6   867         $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                            14   DD  02A6                       PUSHL   #SCB$L_OPCCUS
              00010168 9F   01   FB  02A8                       CALLS   #1, @#DS$CLRVEC
                            56   D4  02AF   868         CLRL    R6                      ; init R6
         56   08   00000507'8F   C1  02B1   869         ADDL3   #TABLE4,#8,R6           ; load R6 with address of exp data
                   00000A48'EF   16  02B9   870         JSB     ERROR_CHECK1            ; go check logout info for errors
           29 00000020'EF   07   E1  02BF   871         BBC     #V_ERROR_FLAG,L_FLAGS,4$ ; br = no error
                                     02C7   872         $DS_ERRHARD_S   UNIT=#0,PRLINK=ERR2_MESSAGE,P1=#T_WCS_ERR,P2=#TABLE4
                   00000507'8F   DD  02C7                       PUSHL   #TABLE4
                   000000AD'8F   DD  02CD                       PUSHL   #T_WCS_ERR
                   000007B6'EF   DF  02D3                       PUSHAL  ERR2_MESSAGE
                            00   DD  02D9                       PUSHL   #0
                            00   DD  02DB                       PUSHL   #0
                            01   DD  02DD                       PUSHL   #$ER
                                     02DF       ::: TEST 8, SUBTEST 3, ERROR 1
              000100D0 9F   06   FB  02DF                       CALLS   #$$M, @#DS$ERRHARD
                                     02E6   873         $DS_BCOMPLETE   4$
                         07 50   E8  02E6                       BLBS    R0, 4$
                                     02E9   874         $DS_ABORT
              00010020 9F   6C   FA  02E9                       CALLG   (AP), @#DS$ABORT
                                     02F0   875 4$:
                                     02F0   876
                                     02F0   877         $DS_ENDSUB
                                     02F0       T8_S3_X:
     00010038 9F   00000018'EF   FA  02F0                       CALLG   $S$, @#DS$ENDSUB
                                     02FB   878
                                     02FB   879         $DS_PAGE
                   000000C0 8F   DD  02FB                       PUSHL   #SCB$L_TIMER
              00010168 9F   01   FB  0301                       CALLS   #1, @#DS$CLRVEC
                       50   01   D0  0308                       MOVL    #1, R0          ; NORMAL EXIT
                                     030B       TEST_008_X::
              00010058 9F   6E   FA  030B                       CALLG   (SP), @#DS$BREAK
                                 04  0312                       RET                     ; RETURN TO TEST SEQUENCER
                                 00000000                       .PSECT  $TSTCNT, NOEXE, NOWRT, OVR, LONG
                           00000008  0000                       .LONG   $TN
                                     0004   883         .END
```

```
$$$                                = 00000018 R      05        DS$GETMEM                              00010130
$$ARGS                             = 0000000A                  DS$GETTERM                             000101C8
$$E                                = 00000001                  DS$GPHARD                              00010018
$$M                                = 00000006                  DS$HELP                                000101B0
$$N                                = 00000001                  DS$INITSCB                             00010170
$$S                                = FFFFFFFF                  DS$INLOOP                              00010048
$$T1                               = 0000002C                  DS$K_ERROR                           = 00000002
$ENV                               = 00000001  G               DS$K_NORMAL                          = 00000001
$ER                                = FFFFFFFF                  DS$K_SEVERE                          = 00000004
$MO                                = 00000001  G               DS$K_SUBSYS                          = 00000066
$S$                                = 00000018 R      05        DS$K_WARNING                         = 00000000
$ST                                = 00000000                  DS$LOAD                                00010198
$TN                                = 00000008                  DS$LOADPCS                             000101C0
ALL                                = 00000009  G               DS$MAPDBGBLOCK                         00010118
A_LAST_ADDRESS                     = FFFFFFFC                  DS$MEMSIZE                             000101D8
A_MCHK_BUFFER                        ********   X    01        DS$MMOFF                               00010158
A_PSLMNE                             ********   X    01        DS$MMON                                00010150
A_WCS_ADDRESS                        00000024 R      01        DS$MOVPHY                              00010148
A_WCS_ENA_ADDR                       00000028 R      01        DS$MOVVRT                              00010140
BASE_008                             00000000 R      03        DS$PARSE                               000100B8
BIT...                             = 0000001A                  DS$PRINTB                              000100E0
BUFSIZ                             = 00000080                  DS$PRINTF                              000100F0
CEP_FUNCTIONAL                     = 00000000                  DS$PRINTS                              000100F8
CEP_REPAIR                         = 00000001                  DS$PRINTSIG                            00010100
CLEAR_BUFFER                         00000AA9 R      01        DS$PRINTX                              000100E8
DATA_008                             0000002C R      03        DS$PROBE                               000101A0
DATA_1                               00000483 R      01        DS$RELBUF                              00010128
DATA_2                               00000487 R      01        DS$RELMEM                              00010138
DEFAULT                            = 00000000  G               DS$SERVICE_DISPATCHER                  000101D0
DS$ABORT                             00010020                  DS$SETIPL                              00010178
DS$ASKADR                            00010090                  DS$SETMAP                              00010188
DS$ASKDATA                           00010080                  DS$SETPRIEXV                           00010110
DS$ASKLGCL                           00010098                  DS$SETVEC                              00010160
DS$ASKSTR                            000100A0                  DS$SHOCHAN                             00010190
DS$ASKVLD                            00010088                  DS$SUMMARY                             00010028
DS$ATTACH                            000101A8                  DS$WAITMS                              00010060
DS$BGNSUB                            00010030                  DS$WAITUS                              00010068
DS$BRANCH                            000100A8                  DS$_ARITH                            = 006600D0
DS$BREAK                             00010058                  DS$_ASBE                             = 00660118
DS$CANWAIT                           00010070                  DS$_BADLINK                          = 006600F0
DS$CHANNEL                           00010180                  DS$_BADTYPE                          = 006600E8
DS$CKLOOP                            00010040                  DS$_BIIC                             = 00660120
DS$CLRVEC                            00010168                  DS$_CHME                             = 006600A8
DS$CNTRLC                            00010078                  DS$_CHMK                             = 006600E0
DS$CVTREG                            000100B0                  DS$_DEVNAME                          = 00660108
DS$ENDPASS                           00010010                  DS$_ERROR                            = 00660002
DS$ENDSUB                            00010038                  DS$_FHWE                             = 00660068
DS$ERRDEV                            000100C8                  DS$_FRAGBUF                          = 00660080
DS$ERRHARD                           000100D0                  DS$_ICBUSY                           = 006600C8
DS$ERRPREP                           00010108                  DS$_ICERR                            = 006600C0
DS$ERRSOFT                           000100D8                  DS$_IHWE                             = 00660060
DS$ERRSYS                            000100C0                  DS$_ILLCHAR                          = 00660018
DS$ESCAPE                            00010050                  DS$_ILLPAGCNT                        = 00660078
DS$FREEDBGSYM                        000101B8                  DS$_ILLUNIT                          = 00660100
DS$GETBUF                            00010120                  DS$_INSFMEM                          = 00660050
```

```
DS$_IPL2HI                     = 006600B8      ERR$_MSGADR                    = 0000000C
DS$_IVADDR                     = 00660040      ERR$_NARGS                     = 0000000A
DS$_IVVECT                     = 00660038      ERR$_NUM                       = 00000004
DS$_KRNLSTK                    = 00660090      ERR$_P1                        = 00000014
DS$_LOGIC                      = 00660070      ERR$_P2                        = 00000018
DS$_MCHK                       = 00660088      ERR$_P3                        = 0000001C
DS$_MMOFF                      = 00660058      ERR$_P4                        = 00000020
DS$_NEEDUNIT                   = 006600F8      ERR$_P5                        = 00000024
DS$_NODE                       = 00660128      ERR$_P6                        = 00000028
DS$_NOPCS                      = 00660110      ERR$_POINTER                   = 00000010
DS$_NORMAL                     = 00660001      ERR$_UNIT                      = 00000008
DS$_NOSUPPORT                  = 006600B1      ERR2_MESSAGE                     000007B6 R      01
DS$_NOTDON                     = 00660030      ERROR_CHECK                      00000927 R      01
DS$_NOTIMP                     = 006600B0      ERROR_CHECK1                     00000A48 R      01
DS$_NULLSTR                    = 00660010      ERR_MESSAGE                      00000543 R      01
DS$_OVERFLOW                   = 00660008      GPR_TABLE                        ********   X    01
DS$_POWER                      = 00660098      HALT                           = 00000004  G
DS$_PROGERR                    = 00660020      HP$A_DEPENDENT                   00000032
DS$_SEVERE                     = 00660004      HP$A_DEVICE                      00000018
DS$_TRANSL                     = 006600A0      HP$A_DVA                         0000001C
DS$_TRUNCATE                   = 00660028      HP$A_LINK                        00000020
DS$_UNEXPINT                   = 006600D8      HP$B_DRIVE                       0000000B
DS$_VASFULL                    = 00660048      HP$B_FLAGS                       0000000A
DS$_WARNING                    = 00660000      HP$Q_DEVICE                      00000000
DSA$AL_APTMAIL                   0000FE00      HP$T_DEVICE                      0000000C
DSA$AT_APTTXT                    0000FA00      HP$T_TYPE                        00000026
DSA$GL_APTCOM                    0000FE04      HP$W_SIZE                        00000008
DSA$GL_DEVLEN                    0000FE58      HP$W_VECTOR                      00000024
DSA$GL_ERRNO                     0000FE44      IPR                            = 00000006  G
DSA$GL_EVENT                     0000FE48      KA$B_ACC_TYPE                    00000042  G
DSA$GL_FLAGS                     0000FE00      KA$B_NODE_ID                     00000046  G
DSA$GL_MSGTYP                    0000FE40      KA$B_RESERVED                    00000043  G
DSA$GL_PASSES                    0000FE08      KA$B_SID_TYPE                    0000003D  G
DSA$GL_PASSNO                    0000FE54      KA$K_LEN                         0000004B  G
DSA$GL_SECTNO                    0000FE10      KA$L_FLAGS                       00000032  G
DSA$GL_SID                       0000FE14      KA$L_MEM_SIZE                    00000047  G
DSA$GL_SUBTNO                    0000FE4C      KA$L_SID                         0000003A  G
DSA$GL_TESTNO                    0000FE50      KA$M_CHAR                      = 00000100  G
DSA$GL_UNITS                     0000FE0C      KA$M_CRC                       = 00000010  G
DSA$GQ_MSGPTR                    0000FE68      KA$M_D_FLOAT                   = 00000002  G
DSA$GT_DEVNAM                    0000FE5C      KA$M_EDIT                      = 00000080  G
ENV$M_DOMAIN                   = 00000002      KA$M_F_FLOAT                   = 00000001  G
ENV$M_LEVEL                    = 00000001      KA$M_G_FLOAT                   = 00000004  G
ENV$M_SUPER                    = 000003FC      KA$M_H_FLOAT                   = 00000008  G
ENV$S_DOMAIN                   = 00000001      KA$M_PACKED                    = 00000020  G
ENV$S_LEVEL                    = 00000001      KA$M_PACK_MUL                  = 00000040  G
ENV$S_SUPER                    = 00000008      KA$M_SB_ERR                    = 02000000  G
ENV$V_DOMAIN                   = 00000001      KA$M_TODR                      = 00010000  G
ENV$V_LEVEL                    = 00000000      KA$M_WCS_LD                    = 01000000  G
ENV$V_SUPER                    = 00000002      KA$V_CHAR                      = 00000008  G
ENV$_CPU                       = 00000000      KA$V_CRC                       = 00000004  G
ENV$_FUNCTIONAL                = 00000000      KA$V_D_FLOAT                   = 00000001  G
ENV$_REPAIR                    = 00000001      KA$V_EDIT                      = 00000007  G
ENV$_SUPER                     = 00000001      KA$V_F_FLOAT                   = 00000000  G
ENV$_SYSTEM                    = 00000001      KA$V_G_FLOAT                   = 00000002  G
```

```
KA$V_H_FLOAT                    = 00000003  G               M_XFC_ERR                        = 00008000
KA$V_PACKED                     = 00000005  G               PAR$M_ATDEF                      = 00000010
KA$V_PACK_MUL                   = 00000006  G               PAR$M_ATHI                       = 00000008
KA$V_SB_ERR                     = 00000019  G               PAR$M_ATLO                       = 00000004
KA$V_TODR                       = 00000010  G               PAR$M_NODEF                      = 00000002
KA$V_WCS_LD                     = 00000018  G               PAR$V_ATDEF                      = 00000004
KA$W_LATENCY                      0000003E  G               PAR$V_ATHI                       = 00000003
KA$W_MEM_SIZE                     00000044  G               PAR$V_ATLO                       = 00000002
KA$W_PROGRESS                     00000040  G               PAR$V_NODEF                      = 00000001
KA$W_RESERVED                     00000038  G               PAR$_BIN                         = 00000002
KA$W_WCS                          00000036  G               PAR$_DEC                         = 0000000A
KA_PTABLE                         *********   X   03        PAR$_HEX                         = 00000010
LOAD_GPR                          00000832 R      01        PAR$_NO                          = 00000000
LOAD_MICRO                        00000AC1 R      01        PAR$_OCT                         = 00000008
L_ACT_ISP                         *********   X   01        PAR$_YES                         = 00000001
L_ACT_KSP                         *********   X   01        PCS                              = 00000008  G
L_ACT_PSL                         *********   X   01        POWER_FAIL                       = 00000003  G
L_ACT_SP                          *********   X   01        PR$_ICCS                           *********   X   01
L_ADDRESS                         00000004 R      01        PR$_ISP                            *********   X   01
L_COUNT                           00000000 R      01        PR$_KSP                            *********   X   01
L_CUR_ISP                         00000008 R      01        PR$_SID                            *********   X   01
L_DATA                          = 00000000                  PR750$_MCESR                     = 00000026
L_EXP_AP                          *********   X   01        PSL$K_EXEC                       = 00000001
L_EXP_FP                          *********   X   01        PSL$K_KERNEL                     = 00000000
L_EXP_ISP                         *********   X   01        PSL$K_SUPER                      = 00000002
L_EXP_KSP                         *********   X   01        PSL$K_USER                       = 00000003
L_EXP_MOD                         00000010 R      01        PSL$M_CBIT                       = 00000001
L_EXP_PSL                         *********   X   01        PSL$M_NBIT                       = 00000008
L_EXP_SP                          *********   X   01        PSL$M_VBIT                       = 00000002
L_EXP_STACK                       00000014 R      01        PSL$M_ZBIT                       = 00000004
L_FLAGS                           00000020 R      01        PSL$S_CURMOD                     = 00000002
L_REV_CHECK                       *********   X   01        PSL$V_CBIT                       = 00000000
L_TIMER                           0000001C R      01        PSL$V_CURMOD                     = 00000018
MANUAL                          = 00000001  G               PSL$V_IS                         = 0000001A
MCHK                            = 00000005  G               PSL$V_NBIT                       = 00000003
MICRO_DATA                        0000048B R      01        PSL$V_VBIT                       = 00000001
MODELIST                          *********   X   01        PSL$V_ZBIT                       = 00000002
MSG_008                           00000002 R      03        READ_GPR                           0000085C R      01
M_BIT19                         = 00080000                  READ_MCHK                          000008B8 R      01
M_BIT20                         = 00100000                  REG_TABLE                          *********   X   01
M_DONE                          = 00000002                  RENTRY3                            000000C2 R      03
M_ERROR_FLAG                    = 00000080                  RENTRY_POINT                       0000000C R      01
M_EXECUTE_ERR                   = 00000004                  SCB$L_ACCESS                       00000020
M_FATAL_ERR                     = 00000010                  SCB$L_ARITH                        00000034
M_GPR_ERR                       = 00000040                  SCB$L_BREAK                        0000002C
M_INT                           = 00002000                  SCB$L_CHME                         00000044
M_ISP_ERR                       = 00000200                  SCB$L_CHMK                         00000040
M_ISTK                          = 00000800                  SCB$L_CHMS                         00000048
M_KSP_ERR                       = 00000100                  SCB$L_CHMU                         0000004C
M_MCINT                         = 00004000                  SCB$L_COMPAT                       00000030
M_MC_ERR                        = 00001000                  SCB$L_KNLSTK                       00000008
M_PSL_ERR                       = 00000020                  SCB$L_MACHCHK                      00000004
M_READY                         = 00000001                  SCB$L_OPCCUS                       00000014
M_SP_ERR                        = 00000400                  SCB$L_OPCDEC                       00000010
M_TIMER                         = 00000008                  SCB$L_POWER                        0000000C
```

```
SCB$L_RADRMOD                   0000001C                SYS$SETPRT                      00010430
SCB$L_ROPRAND                   00000018                SYS$SETRWM                      00010438
SCB$L_RXDB                      000000F8                SYS$SETSWM                      00010448
SCB$L_SFTLVL1                   00000084                SYS$ULKPAG                      00010460
SCB$L_SFTLVL10                  000000A8                SYS$ULWSET                      00010468
SCB$L_SFTLVL11                  000000AC                SYS$UNWIND                      00010470
SCB$L_SFTLVL12                  000000B0                SYS$WAITFR                      00010478
SCB$L_SFTLVL13                  000000B4                SYS$WFLAND                      00010488
SCB$L_SFTLVL14                  000000B8                SYS$WFLOR                       00010490
SCB$L_SFTLVL15                  000000BC                T8_S1                           00000047 RG    03
SCB$L_SFTLVL2                   00000088                T8_S1_X                         0000011B R     03
SCB$L_SFTLVL3                   0000008C                T8_S2                           00000126 RG    03
SCB$L_SFTLVL4                   00000090                T8_S2_X                         00000210 R     03
SCB$L_SFTLVL5                   00000094                T8_S3                           0000021B RG    03
SCB$L_SFTLVL6                   00000098                T8_S3_X                         000002F0 R     03
SCB$L_SFTLVL7                   0000009C                TABLE3                          000004CB R     01
SCB$L_SFTLVL8                   000000A0                TABLE4                          00000507 R     01
SCB$L_SFTLVL9                   000000A4                TAG_3                           000000B4 R     03
SCB$L_TBIT                      00000028                TAG_4                           0000028F R     03
SCB$L_TIMER                     000000C0                TEST_008                        00000030 RG    03
SCB$L_TRANSL                    00000024                TEST_008_X                      0000030B RG    03
SCB$L_TXDB                      000000FC                TIMER_SERVICE                   00000AA0 R     01
SCB$L_WTBUSERR                = 00000060                TUS8                          = 00000002  G
SCB$L_ZERO                      00000000                T_ACT_PSL                       000003BB R     01
SEP_FUNCTIONAL                = 00000002                T_BAD_DATA                      00000309 R     01
SEP_REPAIR                    = 00000003                T_BUSERR                        00000055 R     01
SIZ...                        = 00000001                T_EXECUTE                       0000017C R     01
SYS$ALLOC                       00010238                T_EXPECTED                      00000108 R     01
SYS$ASCTIM                      00010248                T_EXP_PSL                       0000039A R     01
SYS$ASSIGN                      00010250                T_GOOD_DATA                     00000332 R     01
SYS$BINTIM                      00010258                T_GPR                           000003F3 R     01
SYS$CANCEL                      00010260                T_GPR_ERR                       000002E0 R     01
SYS$CANTIM                      00010268                T_ISP_ERR1                      00000245 R     01
SYS$CLOSE                       000105B8                T_ISP_ERR2                      00000279 R     01
SYS$CLREF                       00010298                T_KSP_ERR1                      000001E6 R     01
SYS$CONNECT                     000105C0                T_KSP_ERR2                      00000217 R     01
SYS$DALLOC                      000102D8                T_MC_ERR                        000000D7 R     01
SYS$DASSGN                      000102E0                T_MC_MESS                       0000002C R     01
SYS$DISCONNECT                  000105D0                T_NOT_ON_IS                     00000373 R     01
SYS$FAO                         00010350                T_NOT_ON_KS                     0000034F R     01
SYS$FAOL                        00010358                T_NOT_READY                     0000014A R     01
SYS$GET                         00010580                T_NOXOR                         00000437 R     01
SYS$GETCHN                      000104C8                T_NO_WCS                        0000044D R     01
SYS$GETTIM                      00010378                T_PSL                           000003DA R     01
SYS$LKWSET                      000103A0                T_SP_ERR1                       000002AA R     01
SYS$NUMTIM                      000103B8                T_SP_ERR2                       000002D4 R     01
SYS$OPEN                        00010608                T_TIMER                         000001A1 R     01
SYS$QIO                         000103C8                T_UNEXP_DATA                    00000115 R     01
SYS$QIOW                        00010200                T_WCS_ERR                       000000AD R     01
SYS$READ                        00010590                T_WTBUSERR                      00000080 R     01
SYS$READEF                      000103D0                T_XFC                           000001C9 R     01
SYS$SETAST                      000103F8                T_XOR                           0000041B R     01
SYS$SETEF                       00010400                V_DONE                        = 00000001
SYS$SETIMR                      00010420                V_ERROR_FLAG                  = 00000007
SYS$SETPRI                      00010428                V_EXECUTE_ERR                 = 00000002
```

```
V_FATAL_ERR                     = 00000004
V_GPR_ERR                       = 00000006
V_INT                           = 0000000D
V_ISP_ERR                       = 00000009
V_ISTK                          = 0000000B
V_KSP_ERR                       = 00000008
V_MCINT                         = 0000000E
V_MC_ERR                        = 0000000C
V_PSL_ERR                       = 00000005
V_READY                         = 00000000
V_SP_ERR                        = 0000000A
V_TIMER                         = 00000003
V_XFC_ERR                       = 0000000F
WCS                             = 00000007  G
WTBUS_SERVICE                     000008D4 R      01
XFC_SERVICE                       00000AF8 R      01

                                          +----------------+
                                          ! Psect synopsis !
                                          +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
ECKAXF                          00000B04  ( 2820.)  01 (  1.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC PAGE
$ABS$                           00010610  (67088.)  02 (  2.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
TEST_008                        00000313  (  787.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC PAGE
DISPATCH                        00000018  (   24.)  04 (  4.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD   NOWRT NOVEC LONG
ARGLIST                         00000024  (   36.)  05 (  5.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC LONG
$TSTCNT                         00000004  (    4.)  06 (  6.)  NOPIC   USR   OVR   REL   LCL NOSHR NOEXE   RD   NOWRT NOVEC LONG
```

```
                                    +------------------------+
                                    ! Symbol Cross Reference !
                                    +------------------------+

SYMBOL                 VALUE        DEFINITION     REFERENCES...
-- ---                 -----        ----------     -------------
$$$                    =00000018-R  840      (13)     704    (11)     781    (12)     840    (13)
$$ARGS                 =0000000A    83       (3)       83    (3)
$$E                    -00000001    840      (13)     732    (11)     809    (12)     877    (13)
$$M                    =00000006    872      (13)     654    (10)     726    (11)     803    (12)     872    (13)
$$N                    =00000001    872      (13)   #-339    (8)    #-340    (8)    #-342    (8)    #-346    (8)
                                                    #-349    (8)    #-351    (8)      354    (8)      356    (8)
                                                      359    (8)      361    (8)      363    (8)      365    (8)
                                                      371    (8)      375    (8)      380    (8)    #-382    (8)
                                                      390    (8)      392    (8)    #-395    (8)    #-397    (8)
                                                    #-404    (8)    #-405    (8)    #-406    (8)    #-410    (8)
                                                    #-413    (8)    #-415    (8)    #-417    (8)      726    (11)
                                                      803    (12)   #-847    (13)     872    (13)      88    (3)
$$S                    =FFFFFFFF    881      (13)     645    (10)     654    (10)     659    (11)     704    (11)
                                                      736    (12)     781    (12)     813    (13)     840    (13)
$$T1                   =0000002C    83       (3)       83    (3)
$ENV                   =00000001    66       (3)
$ER                    =FFFFFFFF    881      (13)     704    (11)   #-726    (11)     781    (12)   #-803    (12)
                                                      840    (13)   #-872    (13)
$MO                    =00000001    882      (13)     882    (13)
$S$                    =00000018-R  840      (13)     704    (11)     732    (11)     781    (12)     809    (12)
                                                      840    (13)     877    (13)
$ST                    =00000000    881      (13)     654    (10)     659    (11)     704    (11)     726    (11)
                                                      732    (11)     736    (12)     781    (12)     803    (12)
                                                      809    (12)     813    (13)     840    (13)     848    (13)
                                                      872    (13)     877    (13)     881    (13)
$TN                    =00000008    882      (13)     645    (10)     654    (10)     659    (11)     726    (11)
                                                      736    (12)     803    (12)     813    (13)     872    (13)
                                                      881    (13)     882    (13)
ALL                    =00000009    88       (3)      654    (10)
A_LAST_ADDRESS         =FFFFFFFC    144      (3)    #-717    (11)   #-794    (12)
A_MCHK_BUFFER           00000000-XR                 #-343    (8)    #-384    (8)    #-407    (8)    #-448    (9)
                                                    #-449    (9)    #-476    (9)    #-493    (9)    #-494    (9)
                                                    #-520    (9)    #-577    (9)      609    (9)
A_PSLMNE                00000000-XR                   370    (8)      374    (8)      379    (8)
A_WCS_ADDRESS           00000024-R  148      (3)    #-625    (9)    #-629    (9)    #-863    (13)
A_WCS_ENA_ADDR          00000028-R  150      (3)    #-623    (9)
BASE_008                00000000-R  645      (10)     654    (10)
BIT...                 =0000001A    85       (3)       66    (3)       80    (3)       84    (3)       85    (3)
BUFSIZ                 =00000080    142      (3)    #-337    (8)    #-370    (8)    #-374    (8)    #-379    (8)
CEP_FUNCTIONAL         =00000000    66       (3)
CEP_REPAIR             =00000001    66       (3)       66    (3)
CLEAR_BUFFER            00000AA9-R  608      (9)      705    (11)     782    (12)     852    (13)
DATA_008                0000002C-R  654      (10)     654    (10)
DATA_1                  00000483-R  256      (5)    #-714    (11)
DATA_2                  00000487-R  258      (5)    #-791    (12)
DEFAULT                =00000000    88       (3)      654    (10)
DS$ABORT                00010020                      728    (11)     730    (11)     805    (12)     807    (12)
                                                      874    (13)
```

```
DS$BGNSUB                       00010030                     704    (11)     781    (12)     840    (13)
DS$BREAK                        00010058                     881    (13)
DS$CLRVEC                       00010168                     721    (11)     799    (12)     866    (13)     867    (13)
                                                             880    (13)
DS$CVTREG                       000100B0                     370    (8)      374    (8)      379    (8)
DS$ENDSUB                       00010038                     732    (11)     809    (12)     877    (13)
DS$ERRHARD                      000100D0                     726    (11)     803    (12)     872    (13)
DS$K_ERROR                     =00000002    80     (3)
DS$K_NORMAL                    =00000001    80     (3)
DS$K_SEVERE                    =00000004    80     (3)
DS$K_SUBSYS                    =00000066    80     (3)      80     (3)
DS$K_WARNING                   =00000000    80     (3)
DS$PRINTB                       000100E0                     339    (8)      340    (8)      342    (8)      346    (8)
                                                             349    (8)      351    (8)      354    (8)      359    (8)
                                                             363    (8)      371    (8)      375    (8)      382    (8)
                                                             390    (8)      395    (8)      397    (8)      404    (8)
                                                             405    (8)      406    (8)      410    (8)      413    (8)
                                                             415    (8)      417    (8)
DS$PRINTF                       000100F0                     847    (13)
DS$PRINTSIG                     00010100                     345    (8)      347    (8)      409    (8)      411    (8)
DS$PRINTX                       000100E8                     356    (8)      361    (8)      365    (8)      380    (8)
                                                             392    (8)
DS$SETVEC                       00010160                     655    (10)     707    (11)     784    (12)     853    (13)
                                                             855    (13)
DS$_ARITH                      =006600D0    80     (3)
DS$_ASBE                       =00660118    80     (3)
DS$_BADLINK                    =006600F0    80     (3)
DS$_BADTYPE                    =006600E8    80     (3)
DS$_BIIC                       =00660120    80     (3)
DS$_CHME                       =006600A8    80     (3)
DS$_CHMK                       =006600E0    80     (3)
DS$_DEVNAME                    =00660108    80     (3)
DS$_ERROR                      =00660002    80     (3)
DS$_FHWE                       =00660068    80     (3)
DS$_FRAGBUF                    =00660080    80     (3)
DS$_ICBUSY                     =006600C8    80     (3)
DS$_ICERR                      =006600C0    80     (3)
DS$_IHWE                       =00660060    80     (3)
DS$_ILLCHAR                    =00660018    80     (3)
DS$_ILLPAGCNT                  =00660078    80     (3)
DS$_ILLUNIT                    =00660100    80     (3)
DS$_INSFMEM                    =00660050    80     (3)
DS$_IPL2HI                     =006600B8    80     (3)
DS$_IVADDR                     =00660040    80     (3)
DS$_IVVECT                     =00660038    80     (3)
DS$_KRNLSTK                    =00660090    80     (3)
DS$_LOGIC                      =00660070    80     (3)
DS$_MCHK                       =00660088    80     (3)      298    (7)      315    (7)
DS$_MMOFF                      =00660058    80     (3)
DS$_NEEDUNIT                   =006600F8    80     (3)
DS$_NODE                       =00660128    80     (3)
DS$_NOPCS                      =00660110    80     (3)
DS$_NORMAL                     =00660001    80     (3)
DS$_NOSUPPORT                  =006600B1    80     (3)
DS$_NOTDON                     =00660030    80     (3)
```

```
DS$_NOTIMP                      =006600B0     80       (3)      80      (3)
DS$_NULLSTR                     =00660010     80       (3)
DS$_OVERFLOW                    =00660008     80       (3)
DS$_POWER                       =00660098     80       (3)
DS$_PROGERR                     =00660020     80       (3)
DS$_SEVERE                      =00660004     80       (3)
DS$_TRANSL                      =006600A0     80       (3)
DS$_TRUNCATE                    =00660028     80       (3)
DS$_UNEXPINT                    =006600D8     80       (3)
DS$_VASFULL                     =00660048     80       (3)
DS$_WARNING                     =00660000     80       (3)      80      (3)
DSA$GL_PASSNO                    0000FE54                     #-845     (13)
ENV$M_DOMAIN                    =00000002     66       (3)
ENV$M_LEVEL                     =00000001     66       (3)
ENV$M_SUPER                     =000003FC     66       (3)
ENV$S_DOMAIN                    =00000001     66       (3)
ENV$S_LEVEL                     =00000001     66       (3)
ENV$S_SUPER                     =00000008     66       (3)
ENV$V_DOMAIN                    =00000001     66       (3)      66      (3)
ENV$V_LEVEL                     =00000000     66       (3)      66      (3)
ENV$V_SUPER                     =00000002     66       (3)
ENV$_CPU                        =00000000     66       (3)      66      (3)
ENV$_FUNCTIONAL                 =00000000     66       (3)      66      (3)
ENV$_REPAIR                     =00000001     66       (3)      66      (3)
ENV$_SUPER                      =00000001     66       (3)
ENV$_SYSTEM                     =00000001     66       (3)      66      (3)
ERR$_MSGADR                     =0000000C     83       (3)
ERR$_NARGS                      =0000000A     83       (3)
ERR$_NUM                        =00000004     83       (3)
ERR$_P1                         =00000014     83       (3)     340      (8)       405      (8)
ERR$_P2                         =00000018     83       (3)     347      (8)       411      (8)
ERR$_P3                         =0000001C     83       (3)
ERR$_P4                         =00000020     83       (3)
ERR$_P5                         =00000024     83       (3)
ERR$_P6                         =00000028     83       (3)
ERR$_POINTER                    =00000010     83       (3)
ERR$_UNIT                       =00000008     83       (3)
ERR2_MESSAGE                     000007B6-R  402       (8)     872      (13)
ERROR_CHECK                      00000927-R  517       (9)     724      (11)      801      (12)
ERROR_CHECK1                     00000A48-R  576       (9)     870      (13)
ERR_MESSAGE                      00000543-R  335       (8)     726      (11)      803      (12)
GPR_TABLE                        00000000-XR                   383      (8)       430      (9)       521      (9)
HALT                            =00000004     88       (3)
IPR                             =00000006     88       (3)
KA$W_WCS                         00000036                     #-843     (13)
KA_PTABLE                        00000000-XR                  #-842     (13)
LOAD_GPR                         00000832-R  429       (9)     715      (11)      786      (12)
LOAD_MICRO                       00000AC1-R  618       (9)     851      (13)
L_ACT_ISP                        00000000-XR                  #-359     (8)      #-361     (8)      #-460     (9)      #-505     (9)
                                                              #-547     (9)
L_ACT_KSP                        00000000-XR                  #-354     (8)      #-356     (8)      #-459     (9)      #-504     (9)
                                                              #-543     (9)
L_ACT_PSL                        00000000-XR                  #-374     (8)      #-375     (8)      #-461     (9)      #-506     (9)
                                                               541      (9)      #-550     (9)        555     (9)        560     (9)
L_ACT_SP                         00000000-XR                  #-363     (8)      #-365     (8)      #-458     (9)      #-503     (9)
```

```
                                                              #-538    (9)
L_ADDRESS                      00000004-R   97      (3)
L_COUNT                        00000000-R   96      (3)       #-610    (9)     #-612    (9)
L_CUR_ISP                      00000008-R   98      (3)
L_DATA                        =00000000    145      (3)       #-794    (12)
L_EXP_AP                       00000000-XR                    #-437    (9)
L_EXP_FP                       00000000-XR                    #-438    (9)
L_EXP_ISP                      00000000-XR                    #-359    (8)     #-547    (9)     #-712    (11)    #-789    (12)
L_EXP_KSP                      00000000-XR                    #-354    (8)     #-543    (9)     #-713    (11)    #-790    (12)
L_EXP_MOD                      00000010-R  100      (3)       #-553    (9)     #-555    (9)
L_EXP_PSL                      00000000-XR                    #-370    (8)     #-371    (8)     #-379    (8)     #-380    (8)
                                                              #-550    (9)       553    (9)       558    (9)     #-714    (11)
                                                              #-791    (12)
L_EXP_SP                       00000000-XR                    #-363    (8)     #-538    (9)     #-710    (11)    #-711    (11)
                                                              #-787    (12)    #-788    (12)
L_EXP_STACK                    00000014-R  101      (3)       #-558    (9)     #-560    (9)
L_FLAGS                        00000020-R  105      (3)         341    (8)       348    (8)       350    (8)       352    (8)
                                                                353    (8)       357    (8)       358    (8)       362    (8)
                                                                366    (8)       381    (8)       394    (8)       396    (8)
                                                                412    (8)       414    (8)       416    (8)     #-507    (9)
                                                              #-524    (9)       526    (9)     #-536    (9)     #-540    (9)
                                                              #-542    (9)     #-545    (9)       546    (9)     #-549    (9)
                                                              #-562    (9)     #-563    (9)       564    (9)     #-565    (9)
                                                                566    (9)     #-567    (9)       568    (9)     #-569    (9)
                                                              #-587    (9)       588    (9)     #-589    (9)       590    (9)
                                                              #-591    (9)       592    (9)     #-593    (9)     #-602    (9)
                                                              #-638    (9)     #-709    (11)    #-716    (11)    #-718    (11)
                                                                725    (11)      729    (11)    #-785    (12)    #-793    (12)
                                                                795    (12)    #-797    (12)    #-800    (12)      802    (12)
                                                                806    (12)    #-850    (13)    #-856    (13)    #-857    (13)
                                                                871    (13)
L_REV_CHECK                    00000000-XR                    #-621    (9)
L_TIMER                        0000001C-R  103      (3)       #-792    (12)    #-796    (12)
MANUAL                        =00000001     88      (3)
MCHK                          =00000005     88      (3)         654    (10)
MICRO_DATA                     0000048B-R  267      (7)         626    (9)
MODELIST                       00000000-XR                      370    (8)       374    (8)       379    (8)
MSG_008                        00000002-R  645      (10)        654    (10)
M_BIT19                       =00080000    140      (3)
M_BIT20                       =00100000    141      (3)       #-629    (9)
M_DONE                        =00000002    110      (3)
M_ERROR_FLAG                  =00000080    122      (3)       #-524    (9)     #-536    (9)     #-540    (9)     #-545    (9)
                                                              #-549    (9)     #-563    (9)     #-565    (9)     #-567    (9)
                                                              #-569    (9)     #-587    (9)     #-589    (9)     #-591    (9)
                                                              #-593    (9)
M_EXECUTE_ERR                 =00000004    112      (3)       #-718    (11)    #-797    (12)
M_FATAL_ERR                   =00000010    116      (3)       #-562    (9)
M_GPR_ERR                     =00000040    120      (3)       #-524    (9)
M_INT                         =00002000    134      (3)       #-800    (12)
M_ISP_ERR                     =00000200    126      (3)       #-549    (9)
M_ISTK                        =00000800    130      (3)       #-542    (9)
M_KSP_ERR                     =00000100    124      (3)       #-545    (9)
M_MCINT                       =00004000    136      (3)       #-507    (9)
M_MC_ERR                      =00001000    132      (3)       #-536    (9)     #-587    (9)
M_PSL_ERR                     =00000020    118      (3)       #-563    (9)
```

```
M_READY                          =00000001     108     (3)     #-716     (11)    #-793     (12)    #-857     (13)
M_SP_ERR                         =00000400     128     (3)     #-540     (9)
M_TIMER                          =00000008     114     (3)     #-602     (9)
M_XFC_ERR                        =00008000     138     (3)     #-638     (9)
PAR$M_ATDEF                      =00000010     84      (3)
PAR$M_ATHI                       =00000008     84      (3)
PAR$M_ATLO                       =00000004     84      (3)
PAR$M_NODEF                      =00000002     84      (3)
PAR$V_ATDEF                      =00000004     84      (3)
PAR$V_ATHI                       =00000003     84      (3)
PAR$V_ATLO                       =00000002     84      (3)
PAR$V_NODEF                      =00000001     84      (3)
PAR$_BIN                         =00000002     84      (3)
PAR$_DEC                         =0000000A     84      (3)
PAR$_HEX                         =00000010     84      (3)
PAR$_NO                          =00000000     84      (3)
PAR$_OCT                         =00000008     84      (3)
PAR$_YES                         =00000001     84      (3)
PCS                              =00000008     88      (3)
POWER_FAIL                       =00000003     88      (3)
PR$_ICCS                          00000000-XR                  #-601     (9)
PR$_ISP                           00000000-XR                  #-460     (9)     #-505     (9)     #-712     (11)    #-789     (12)
PR$_KSP                           00000000-XR                  #-459     (9)     #-504     (9)
PR$_SID                           00000000-XR                  #-619     (9)
PR750$_MCESR                     =00000026                     #-706     (11)    #-783     (12)    #-854     (13)
PSL$K_EXEC                       =00000001     85      (3)
PSL$K_KERNEL                     =00000000     85      (3)
PSL$K_SUPER                      =00000002     85      (3)
PSL$K_USER                       =00000003     85      (3)
PSL$M_CBIT                       =00000001     85      (3)
PSL$M_NBIT                       =00000008     85      (3)
PSL$M_VBIT                       =00000002     85      (3)
PSL$M_ZBIT                       =00000004     85      (3)
PSL$S_CURMOD                     =00000002                     #-552     (9)     #-554     (9)
PSL$V_CBIT                       =00000000     85      (3)
PSL$V_CURMOD                     =00000018                     #-552     (9)     #-554     (9)
PSL$V_IS                         =0000001A                     #-541     (9)     #-557     (9)     #-559     (9)
PSL$V_NBIT                       =00000003     85      (3)
PSL$V_VBIT                       =00000001     85      (3)
PSL$V_ZBIT                       =00000002     85      (3)
READ_GPR                          0000085C-R   447     (9)       707     (11)
READ_MCHK                         000008B8-R   475     (9)       855     (13)
REG_TABLE                         00000000-XR                    387     (8)
RENTRY3                           000000C2-R   720     (11)    #-708     (11)
RENTRY_POINT                      0000000C-R   99      (3)       466     (9)     #-708     (11)
SCB$L_MACHCHK                     00000004                     #-707     (11)    #-721     (11)    #-855     (13)    #-866     (13)
SCB$L_OPCCUS                      00000014                     #-853     (13)    #-867     (13)
SCB$L_TIMER                       000000C0                     #-655     (10)    #-880     (13)
SCB$L_WTBUSERR                   =00000060     143     (3)     #-784     (12)    #-799     (12)
SEP_FUNCTIONAL                   =00000002     66      (3)
SEP_REPAIR                       =00000003     66      (3)
SIZ...                           =00000001     85      (3)       66      (3)       84      (3)       85      (3)
T8_S1                             00000047-R   704     (11)
T8_S1_X                           0000011B-R   732     (11)
T8_S2                             00000126-R   781     (12)
```

```
T8_S2_X                         00000210-R   809    (12)
T8_S3                           0000021B-R   840    (13)
T8_S3_X                         000002F0-R   877    (13)   #-848    (13)
TABLE3                          000004CB-R   297    (7)    #-344    (8)     #-408    (8)     #-723    (11)   #-726    (11)
TABLE4                          00000507-R   314    (7)    #-869    (13)   #-872    (13)
TAG_3                           000000B4-R   717    (11)     302    (7)       310    (7)
TAG_4                           0000028F-R   858    (13)     319    (7)       327    (7)
TEST_008                        00000030-R   654    (10)     654    (10)
TEST_008_X                      0000030B-R   881    (13)
TIMER_SERVICE                   00000AA0-R   600    (9)      655    (10)
TU58                           =00000002     88     (3)
T_ACT_PSL                       000003BB-R   230    (4)      375    (8)
T_BAD_DATA                      00000309-R   215    (4)
T_BUSERR                        00000055-R   163    (4)    #-726    (11)
T_EXECUTE                       0000017C-R   185    (4)      351    (8)
T_EXPECTED                      00000108-R   176    (4)      346    (8)       410    (8)
T_EXP_PSL                       0000039A-R   227    (4)      371    (8)
T_GOOD_DATA                     00000332-R   218    (4)
T_GPR                           000003F3-R   236    (4)      382    (8)
T_GPR_ERR                       000002E0-R   212    (4)
T_ISP_ERR1                      00000245-R   200    (4)      359    (8)
T_ISP_ERR2                      00000279-R   203    (4)      361    (8)
T_KSP_ERR1                      000001E6-R   194    (4)      354    (8)
T_KSP_ERR2                      00000217-R   197    (4)      356    (8)
T_MC_ERR                        000000D7-R   172    (4)      342    (8)       406    (8)
T_MC_MESS                       0000002C-R   160    (4)      339    (8)       404    (8)
T_NOT_ON_IS                     00000373-R   224    (4)
T_NOT_ON_KS                     0000034F-R   221    (4)
T_NOT_READY                     0000014A-R   182    (4)      349    (8)       413    (8)
T_NOXOR                         00000437-R   242    (4)      392    (8)
T_NO_WCS                        0000044D-R   245    (4)      847    (13)
T_PSL                           000003DA-R   233    (4)      380    (8)
T_SP_ERR1                       000002AA-R   206    (4)      363    (8)
T_SP_ERR2                       000002D4-R   209    (4)      365    (8)
T_TIMER                         000001A1-R   188    (4)      395    (8)       415    (8)
T_UNEXP_DATA                    00000115-R   179    (4)
T_WCS_ERR                       000000AD-R   169    (4)    #-872    (13)
T_WTBUSERR                      00000080-R   166    (4)    #-803    (12)
T_XFC                           000001C9-R   191    (4)      397    (8)       417    (8)
T_XOR                           0000041B-R   239    (4)      390    (8)
V_DONE                         =00000001     109    (3)
V_ERROR_FLAG                   =00000007     121    (3)    #-725    (11)   #-802    (12)   #-871    (13)
V_EXECUTE_ERR                  =00000002     111    (3)    #-350    (8)    #-566    (9)
V_FATAL_ERR                    =00000004     115    (3)    #-729    (11)   #-806    (12)
V_GPR_ERR                      =00000006     119    (3)    #-381    (8)
V_INT                          =0000000D     133    (3)    #-526    (9)
V_ISP_ERR                      =00000009     125    (3)    #-358    (8)
V_ISTK                         =0000000B     129    (3)    #-352    (8)    #-357    (8)    #-546    (9)
V_KSP_ERR                      =00000008     123    (3)    #-353    (8)
V_MCINT                        =0000000E     135    (3)    #-795    (12)
V_MC_ERR                       =0000000C     131    (3)    #-341    (8)
V_PSL_ERR                      =00000005     117    (3)    #-366    (8)
V_READY                        =00000000     107    (3)    #-348    (8)    #-412    (8)    #-564    (9)    #-588    (9)
V_SP_ERR                       =0000000A     127    (3)    #-362    (8)
V_TIMER                        =00000003     113    (3)    #-394    (8)    #-414    (8)    #-568    (9)    #-592    (9)
```

```
V_XFC_ERR                  =0000000F     137    (3)    #-396    (8)    #-416    (8)    #-590    (9)
WCS                        =00000007     88     (3)
WTBUS_SERVICE               000008D4-R   492    (9)      784    (12)
XFC_SERVICE                 00000AF8-R   637    (9)      320    (7)      853    (13)
```

```
                                           +-----------------------+
                                           ! Macros Cross Reference !
                                           +-----------------------+

MACRO            SIZE         DEFINITION      REFERENCES...
-----            ----         ----------      -------------
$D1_ABPROGRAM    1            728    (11)     728    (11)     730    (11)     805    (12)     807    (12)     874    (13)
$D1_BSUBT        1            704    (11)     704    (11)     781    (12)     840    (13)
$D1_BTEST        1            654    (10)     654    (10)
$D1_ERR          1            726    (11)     726    (11)     803    (12)     872    (13)
$D1_ERR_S        1            726    (11)     726    (11)     803    (12)     872    (13)
$D1_ESUBT        1            732    (11)     732    (11)     809    (12)     877    (13)
$D1_ETEST        1            881    (13)     881    (13)
$D1_EXSUB        1            848    (13)     848    (13)
$D1_PRINT_S      1            339    (8)      339    (8)      340    (8)      342    (8)      346    (8)      349    (8)
                                              351    (8)      354    (8)      356    (8)      359    (8)      361    (8)
                                              363    (8)      365    (8)      371    (8)      375    (8)      380    (8)
                                              382    (8)      390    (8)      392    (8)      395    (8)      397    (8)
                                              404    (8)      405    (8)      406    (8)      410    (8)      413    (8)
                                              415    (8)      417    (8)      847    (13)
$D1_SBTTL        2            645    (10)     645    (10)     659    (11)     736    (12)     813    (13)
$D2_BDATA        1            654    (10)     654    (10)
$D2_BTEST        1            654    (10)     654    (10)
$D2_SBTTL        1            645    (10)     645    (10)
$DEF             1            85     (3)
$DEFINI          1            79     (3)      79     (3)      81     (3)      82     (3)      85     (3)      86     (3)
                                              87     (3)      89     (3)
$DS_ABORT        1            728    (11)     728    (11)     730    (11)     805    (12)     807    (12)     874    (13)
$DS_BCOMPLETE    1            727    (11)     727    (11)     804    (12)     873    (13)
$DS_BGNMESSAGE   1            336    (8)      336    (8)      403    (8)
$DS_BGNMOD       1            66     (3)      66     (3)
$DS_BGNSUB       1            704    (11)     704    (11)     781    (12)     840    (13)
$DS_BGNTEST      1            654    (10)     654    (10)
$DS_BREAK        1            881    (13)     881    (13)
$DS_CLRVEC_S     1            721    (11)     721    (11)     799    (12)     866    (13)     867    (13)     880    (13)
$DS_CVTREG_S     1            368    (8)      368    (8)      372    (8)      377    (8)
$DS_DSADEF       5            81     (3)      81     (3)
$DS_DSDEF        2            80     (3)      80     (3)
$DS_DSSDEF       8            82     (3)      82     (3)
$DS_ENDDATA      1            654    (10)     654    (10)
$DS_ENDMESSAGE   1            399    (8)      399    (8)      419    (8)
$DS_ENDMOD       1            882    (13)     882    (13)
$DS_ENDSUB       1            732    (11)     732    (11)     809    (12)     877    (13)
$DS_ENDTEST      1            881    (13)     881    (13)
$DS_ENVDEF       2            66     (3)      66     (3)
$DS_ERRDEF       1            83     (3)      83     (3)
$DS_ERRHARD_S    1            726    (11)     726    (11)     803    (12)     872    (13)
$DS_EXIT         1            848    (13)     848    (13)
$DS_HPODEF       2            86     (3)      86     (3)
$DS_KA_DEF       3            87     (3)      87     (3)
$DS_PAGE         1            643    (9)      643    (9)      657    (10)     734    (11)     810    (12)     879    (13)
$DS_PARDEF       1            84     (3)      84     (3)
$DS_PRINTB_S     1            339    (8)      339    (8)      340    (8)      342    (8)      346    (8)      349    (8)
                                              351    (8)      354    (8)      359    (8)      363    (8)      371    (8)
```

```
                                                   375       (8)       382       (8)       390       (8)       395       (8)       397       (8)
                                                   404       (8)       405       (8)       406       (8)       410       (8)       413       (8)
                                                   415       (8)       417       (8)
$DS_PRINTF_S         1                847      (13)      847       (13)
$DS_PRINTSIG_G       1                345      (8)       345       (8)       347       (8)       409       (8)       411       (8)
$DS_PRINTX_S         1                356      (8)       356       (8)       361       (8)       365       (8)       380       (8)       392       (8)
$DS_PSLDEF           1                85       (3)       85        (3)
$DS_SBTTL            1                645      (10)      645       (10)      659       (11)      736       (12)      813       (13)
$DS_SCBDEF           3                79       (3)       79        (3)
$DS_SECDEF           1                88       (3)       88        (3)
$DS_SETVEC_S         1                655      (10)      655       (10)      707       (11)      784       (12)      853       (13)      855       (13)
$EQU                 1                85       (3)       66        (3)       80        (3)       84        (3)       85        (3)
$EQULS1              1                85       (3)       66        (3)       80        (3)       85        (3)
$EQULST              1                66       (3)       66        (3)       80        (3)       85        (3)
$GBLINI              2                66       (3)       66        (3)       80        (3)       84        (3)       85        (3)
$OFFDEF              1                83       (3)       83        (3)
$PR750DEF            1                89       (3)       89        (3)
$PSLDEF              2                85       (3)       85        (3)
$PUSHADR             1                370      (8)       370       (8)       374       (8)       379       (8)       726       (11)      803       (12)
                                                         872       (13)
$VIELD               1                66       (3)       66        (3)       84        (3)       85        (3)
$VIELD1              1                85       (3)       66        (3)       84        (3)       85        (3)
ECKAX_SECDEF         1                88       (3)       88        (3)

                                                  +-------------------------+
                                                  ! Performance indicators !
                                                  +-------------------------+

Phase                      Page faults     CPU Time        Elapsed Time
-----                      -----------     --------        ------------
Initialization                   30        00:00:00.09     00:00:00.27
Command processing              897        00:00:00.64     00:00:02.26
Pass 1                         1220        00:00:12.45     00:00:39.28
Symbol table sort                 0        00:00:00.60     00:00:01.19
Pass 2                           31        00:00:02.87     00:00:08.71
Symbol table output               2        00:00:00.24     00:00:01.32
Psect synopsis output             2        00:00:00.02     00:00:00.02
Cross-reference output            7        00:00:01.14     00:00:04.24
Assembler run totals           2191        00:00:18.05     00:00:57.31

The working set limit was 2762 pages.
169085 bytes (331 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 570 non-local and 61 local symbols.
883 source lines were read in Pass 1, producing 33 object records in Pass 2.
130 pages of virtual memory were used to define 57 macros.
```

```
                                           +---------------------------+
                                           ! Macro library statistics !
                                           +---------------------------+

Macro library name                          Macros defined
------------------                          --------------
DISK$DIAGPACK02:[TAYLOR.ECKAX]ECKAX.MLB;1              1
SYS$COMMON:[SYSLIB]DIAG.MLB;955                       47
SYS$COMMON:[SYSLIB]STARLET.MLB;2                       9
TOTALS (all libraries)                                57

895 GETS were required to define 57 macros.

There were no errors, warnings or information messages.

MACRO/LIS=ECKAX6.LIS/CROSS ECKAX6
```

Table of contents

```
    (1)       1        ECKAX7: TU58 EXERCISER
    (2)      67        DECLARATIONS
    (5)     350        COMMAND PACKET TABLE
    (40)   2131        END PACKET TABLE
    (47)   2474        DATA PACKET TABLE
    (52)   2699        COMMAND TABLES
    (53)   3122        ERROR MESSAGES
    (54)   3334        SUBROUTINES
    (56)   3649        TEST 9:  TU58 EXERCISER
    (57)   3715        SUBTEST 9.1:   INITIALIZE
    (58)   3819        SUBTEST 9.2:   SELF DIAGNOSTIC
    (59)   3921        SUBTEST 9.3:   NOP
    (60)   4026        TEST 10:  TU58 EXERCISER READ/WRITE
    (61)   4106        SUBTEST 10.1:   READ DIRECTORY
    (62)   4362        SUBTEST 10.2:   WRITE BLOCK
    (63)   4523        SUBTEST 10.3:   READ BLOCK
    (64)   4652        SUBTEST 10.4:   WRITE MODIFIED
    (65)   4811        SUBTEST 10.5:   READ MODIFIED
    (66)   4938        SUBTEST 10.6:   WRITE RECORD
    (67)   5096        SUBTEST 10.7:   READ RECORD
    (68)   5217        SUBTEST 10.8:   WRITE BLOCK - PADDED ZEROS
    (69)   5501        SUBTEST 10.9:   WRITE RECORD - PADDED ZEROS
    (70)   5775        SUBTEST 10.10:   BLOCK WRITE - READ
    (71)   6050        SUBTEST 10.11:   RECORD WRITE - READ
    (72)   6318        SUBTEST 10.12:   WRITE RECORD INTERRUPT
    (73)   6490        SUBTEST 10.13:   READ RECORD INTERRUPT
    (74)   6619        SUBTEST 10.14:  IPL CHECK
    (76)   6738        TEST 11:  TU58 EXERCISER (WITH MRSP)
    (77)   6804        SUBTEST 11.1:   INITIALIZE
    (78)   6908        SUBTEST 11.2:   SELF DIAGNOSTIC
    (79)   7010        SUBTEST 11.3:   NOP
    (80)   7115        TEST 12:  TU58 EXERCISER READ/WRITE (WITH MRSP)
    (81)   7195        SUBTEST 12.1:   READ DIRECTORY
    (82)   7451        SUBTEST 12.2:   WRITE BLOCK
    (83)   7612        SUBTEST 12.3:   READ BLOCK
    (84)   7741        SUBTEST 12.4:   WRITE MODIFIED
    (85)   7900        SUBTEST 12.5:   READ MODIFIED
    (86)   8027        SUBTEST 12.6:   WRITE RECORD
    (87)   8185        SUBTEST 12.7:   READ RECORD
    (88)   8306        SUBTEST 12.8:   WRITE BLOCK - PADDED ZEROS
    (89)   8590        SUBTEST 12.9:   WRITE RECORD - PADDED ZEROS
    (90)   8864        SUBTEST 12.10:   BLOCK WRITE - READ
    (91)   9139        SUBTEST 12.11:   RECORD WRITE - READ
    (92)   9407        SUBTEST 12.12:   WRITE RECORD INTERRUPT
    (93)   9579        SUBTEST 12.13:   READ RECORD INTERRUPT
    (94)   9707        SUBTEST 12.14:  IPL CHECK
```

```
0000                .SBTTL  ECKAX7: TU58 EXERCISER
0000                .TITLE  ECKAX7  VAX-11/750 Specific CPU Cluster Exerciser
0000                .IDENT  /5.2/
0000
0000          ;
0000          ; Copyright (C) 1980,1983,1986
0000          ; Digital Equipment Corporation, Maynard, Massachusetts 01754
0000          ;
0000          ; This software is furnished  under a license for use only on a single
0000          ; computer  system and  may be  copied only with  the inclusion of the
0000          ; above copyright notice.  This software, or any other copies thereof,
0000          ; may not be provided or  otherwise made available to any other person
0000          ; except for use on such system and to one who agrees to these license
0000          ; terms.  Title to and  ownership of the  software  shall at all times
0000          ; remain in DEC.
0000          ;
0000          ; The information in this software is subject to change without notice
0000          ; and should  not be construed  as a commitment  by Digital  Equipment
0000          ; Corporation.
0000          ;
0000          ; DEC assumes  no  responsibility  for  the use or  reliability of its
0000          ; software on equipment which is not supplied by DEC.
0000          ;
0000
0000          ;++
0000          ; Facility:     VAX Architecture Diagnostic.
0000          ;
0000          ; Abstract:     This module exercises the TU58 tape controller to assure
0000          ;               basic functionality. A series of subtests, including the
0000          ;               controller's self test are executed checking initialize,
0000          ;               no operation, write, read, seek, write modified, read
0000          ;               modified, head switching and positioning in RSP and MRSP.
0000          ;
0000          ;
0000          ; Environment:  VAX Diagnostic Supervisor.
0000          ;
0000          ; Author:       Fred Roemer     17-Feb-81       Version 3.1
0000          ;
0000          ;       NO CHANGES TO THIS MODULE FOR VERSION 3.3 (DOCUMENT CHANGE ONLY)
0000          ;
0000          ;
0000          ;       Dan Milleville  19-Mar-82               Version 3.4
0000          ;       Changed Test 3 exit code to accomodate temporarily a hardware problem.
0000          ;
0000          ;       Dave Shull      30-Nov-82               Version 4.0
0000          ;       Added tests 13 and 14 for coverage of PCS module.
0000          ;
0000          ;       Tai-Chun Pan    04-Mar-83               Version 5.0
0000          ;       Added test 5 for coverage of Processor Halts Architecture Undefined.
0000          ;       Added tests 11 and 12 for coverage of new ROM with MRSP of TU58.
0000          ;
0000          ;       Dan Milleville  21-Nov-83               Version 5.1
0000          ;       Fixed error in timer routine (didn't turn off timer).
0000          ;
0000          ;       Wayne R. Taylor February 1986           Version 5.2
```

```
                            0000     56 ;       Redefined the IPR's to jive with the starlet library.
                            0000     57 ;       Modified the code for problem report Eckax5 where:
                            0000     58 ;               Tests 6 & 13 tests with cache off.
                            0000     59 ;               Subtest 1 does not check bit 0 of the memory under test.
                            0000     60 ;               Subtest 4 tests row 0, 1, 2, but not row 3.
                            0000     61 ;       Modifications were supplied by the submitter.
                            0000     62 ;       Added $PR750DEF to the macro spec.
                            0000     63 ;       Search on 5.2.
                            0000     64 ;
                            0000     65 ;--
```

```
                    0000     67 .SBTTL  DECLARATIONS
                    0000     68         .LIST   MEB
                    0000     69         .NLIST  CND
                00000000     70         .PSECT  ECKAXG, PAGE, NOWRT
                    0000     71         .LIBRARY        "SYS$LIBRARY:DIAG ; VAX family diagnostic library.
                    0000     72         $DS_BGNMOD      CEP_REPAIR, TN=9
                    0000        ;;;     Macro-32 Diagnostic macro library Version 8.3  DIAG.MLB (245)
                    0000     73
                    0000     74 ;
                    0000     75 ; Include files:
                    0000     76 ;
                    0000     77
                    0000     78         .LIBRARY        'ECKAX'         ; CPU Cluster Exerciser library.
                    0000     79
                    0000     80 ;
                    0000     81 ; Macros:
                    0000     82 ;
                    0000     83
                    0000     87
                    0000     88         $PR750DEF       ; 5.2
                    0000     89         $DS_ERRDEF
                    0000     90         $DS_DSDEF
                    0000     91         $DS_DSSDEF
                    0000     92         $DS_PARDEF
                    0000     93         $DS_SCBDEF
                    0000     94         ECKAX_SECDEF
                    0000     95
                    0000     96 ;
                    0000     97 ; Equated symbols:
                    0000     98 ;
                    0000     99
                    0000    100 ; packet flags
                    0000    101
      00000001      0000    102 DATA_FLAG=^X01
      00000002      0000    103 COMMAND_FLAG=^X02
      00000002      0000    104 END_FLAG=^X02
      00000004      0000    105 INIT_FLAG=^X04
      00000010      0000    106 CONTINUE_FLAG=^X10
      00000011      0000    107 XON_FLAG=^X11
      00000013      0000    108 XOFF_FLAG=^X13
                    0000    109
                    0000    110 ; op codes
                    0000    111
      00000000      0000    112 NO_OPERATION=^X00
      00000001      0000    113 INITIALIZE=^X01
      00000002      0000    114 READ=^X02
      00000003      0000    115 WRITE=^X03
      00000005      0000    116 POSITION=^X05
      00000007      0000    117 DIAGNOSE=^X07
      00000040      0000    118 END_CODE=^X40
                    0000    119
                    0000    120 ; packet arguments
                    0000    121
      00000000      0000    122 SUCCESS_CODE=^X00
      00000000      0000    123 MODIFIER_00=^X00                ; Normal operation, no modifier
```

```
        00000001  0000    124 MODIFIER_01=^X01                    ; Write check modifier
        00000080  0000    125 MODIFIER_80=^X80                    ; Record addressing modifier
        00000000  0000    126 SWITCH_00=^X00
        00000008  0000    127 SWITCH_08=^X08
        00000000  0000    128 SEQUENCE_H=^X00
        00000000  0000    129 SEQUENCE_L=^X00
        00000000  0000    130 NOT_USED=^X00
        00000000  0000    131 UNIT_0=^X00
        00000000  0000    132 SUMMARY_STAT_H=^X00
        00000000  0000    133 SUMMARY_STAT_L=^X00
                  0000    134
                  0000    135 ; bit positions and masks
                  0000    136
        00000007  0000    137 V_DONE=7
        00000007  0000    138 V_READY=7
        00000004  0000    139 V_TRANSFER=4
        00000010  0000    140 M_TRANSFER=1@4
        00000040  0000    141 M_ENABLE=1@6
        00000000  0000    142 V_RUN=0
        00000001  0000    143 M_RUN=1@0
        80000000  0000    144 M_ERROR=1@31
        00000080  0000    145 M_INTERRUPT=1@7
        00000004  0000    146 V_RESPONSE_FLAG=4
                  0000    147
                  0000    148 ;PR$_CSTS=^X1E                       ; TU58 transmitter command/status register  5.2
                  0000    149 ;PR$_CSTD=^X1F                       ; TU58 transmitter data register            5.2
                  0000    150 ;PR$_CSRS=^X1C                       ; TU58 receiver command/status register     5.2
                  0000    151 ;PR$_CSRD=^X1D                       ; TU58 receiver data register               5.2
        000000F4  0000    152 SCB$L_CSTD=^XF4
        000000F0  0000    153 SCB$L_CSRD=^XF0
                  0000    154
        00000001  0000    155 B_FLAGS:          .BLKB 1           ; contains flags
        00000002  0001    156 B_FLAGS2:         .BLKB 1
        00000003  0002    157 B_FLAGS3:         .BLKB 1
        00000004  0003    158 B_COUNT:          .BLKB 1           ; contains count when repeating
        00000005  0004    159 B_COUNT1:         .BLKB 1
        00000006  0005    160 B_COUNT2:         .BLKB 1
        0000000A  0006    161 L_COUNT:          .BLKL 1
        0000000E  000A    162 L_COUNT1:         .BLKL 1
        00000012  000E    163 L_COUNT3:         .BLKL 1
        00000013  0012    164 B_RESPONSE:       .BLKB 1           ; contains operator responses
        00000017  0013    165 BUF_LIM:          .BLKL 1
        0000001B  0017    166 TEMP5:            .BLKL 1
        0000001F  001B    167 TEMP7:            .BLKL 1
                  001F    168
                  001F    169 ; bit positions and masks for flags
                  001F    170
        00000000  001F    171 V_CLOCK_FLAG=0
        00000001  001F    172 M_CLOCK_FLAG=1@0
        00000001  001F    173 V_RECEIVE_FLAG=1
        00000002  001F    174 M_RECEIVE_FLAG=1@1
        00000002  001F    175 V_DATA_FLAG=2
        00000004  001F    176 M_DATA_FLAG=1@2
        00000003  001F    177 V_END_FLAG=3
        00000008  001F    178 M_END_FLAG=1@3
```

```
                        00000004  001F    179 V_SEND_FLAG=4
                        00000010  001F    180 M_SEND_FLAG=1@4
                        00000001  001F    181 V_REPEAT_SEND=1
                        00000002  001F    182 M_REPEAT_SEND=1@1
                        00000002  001F    183 V_REPEAT_CHECK=2
                        00000004  001F    184 M_REPEAT_CHECK=1@2
                        00000000  001F    185 V_DONE_FLAG=0
                        00000002  001F    186 M_SDONE_FLAG=1@1
                        00000001  001F    187 V_SDONE_FLAG=1
                        00000004  001F    188 M_RD_INT_FLAG=1@2
                        00000002  001F    189 V_RD_INT_FLAG=2
                                  001F    190
                                  001F    191
                        00000600  001F    192 BUFFER_SIZE=1536                ; buffer size to clear buffers
                                  001F    193
                                  001F    194 ; delay times for send, receive, and wait timers
                                  001F    195
                        FFFB6C20  001F    196 DELAY_TIME1= -300000
                        FFF85EE0  001F    197 DELAY_TIME2= -500000
                        FFF3CB00  001F    198 DELAY_TIME3= -800000
                        FFF0BDC0  001F    199 DELAY_TIME4=-1000000
                        FFE17B80  001F    200 DELAY_TIME5=-2000000
                        FFC2F700  001F    201 DELAY_TIME6=-4000000
                        FF7E4CE0  001F    202 DELAY_TIME7=-8500000
                        FAEF00C0  001F    203 DELAY_TIME8=-85000000
```

```
                     001F     205 ; Checksum list
                     001F     206
           00000001  001F     207 CHECKSUM_01=^X01
           00000002  001F     208 CHECKSUM_02=^X02
           00000003  001F     209 CHECKSUM_03=^X03
           00000004  001F     210 CHECKSUM_04=^X04
           00000005  001F     211 CHECKSUM_05=^X05
           00000008  001F     212 CHECKSUM_08=^X08
           00000009  001F     213 CHECKSUM_09=^X09
           0000000A  001F     214 CHECKSUM_0A=^X0A
           0000000B  001F     215 CHECKSUM_0B=^X0B
           00000001  001F     216 CHECKSUM_01=^X01
           0000000C  001F     217 CHECKSUM_0C=^X0C
           0000000D  001F     218 CHECKSUM_0D=^X0D
           0000000E  001F     219 CHECKSUM_0E=^X0E
           0000000F  001F     220 CHECKSUM_0F=^X0F
           00000010  001F     221 CHECKSUM_10=^X10
           00000011  001F     222 CHECKSUM_11=^X11
           00000012  001F     223 CHECKSUM_12=^X12
           00000013  001F     224 CHECKSUM_13=^X13
           00000014  001F     225 CHECKSUM_14=^X14
           00000015  001F     226 CHECKSUM_15=^X15
           00000016  001F     227 CHECKSUM_16=^X16
           00000019  001F     228 CHECKSUM_19=^X19
           0000001A  001F     229 CHECKSUM_1A=^X1A
           0000001C  001F     230 CHECKSUM_1C=^X1C
           0000001D  001F     231 CHECKSUM_1D=^X1D
           0000002D  001F     232 CHECKSUM_2D=^X2D
           00000036  001F     233 CHECKSUM_36=^X36
           00000041  001F     234 CHECKSUM_41=^X41
           00000042  001F     235 CHECKSUM_42=^X42
           00000043  001F     236 CHECKSUM_43=^X43
           00000054  001F     237 CHECKSUM_54=^X54
           0000005C  001F     238 CHECKSUM_5C=^X5C
           0000006E  001F     239 CHECKSUM_6E=^X6E
           0000006F  001F     240 CHECKSUM_6F=^X6F
           00000072  001F     241 CHECKSUM_72=^X72
           00000075  001F     242 CHECKSUM_75=^X75
           00000079  001F     243 CHECKSUM_79=^X79
           00000080  001F     244 CHECKSUM_80=^X80
           00000082  001F     245 CHECKSUM_82=^X82
           00000083  001F     246 CHECKSUM_83=^X83
           00000084  001F     247 CHECKSUM_84=^X84
           00000085  001F     248 CHECKSUM_85=^X85
           00000088  001F     249 CHECKSUM_88=^X88
           00000089  001F     250 CHECKSUM_89=^X89
           0000008A  001F     251 CHECKSUM_8A=^X8A
           0000008B  001F     252 CHECKSUM_8B=^X8B
           0000008C  001F     253 CHECKSUM_8C=^X8C
```

```
0000008D  001F    255 CHECKSUM_8D=^X8D
00000090  001F    256 CHECKSUM_90=^X90
00000091  001F    257 CHECKSUM_91=^X91
00000092  001F    258 CHECKSUM_92=^X92
00000093  001F    259 CHECKSUM_93=^X93
00000094  001F    260 CHECKSUM_94=^X94
00000095  001F    261 CHECKSUM_95=^X95
00000098  001F    262 CHECKSUM_98=^X98
000000A6  001F    263 CHECKSUM_A6=^XA6
000000A9  001F    264 CHECKSUM_A9=^XA9
000000AA  001F    265 CHECKSUM_AA=^XAA
000000AB  001F    266 CHECKSUM_AB=^XAB
000000AE  001F    267 CHECKSUM_AE=^XAE
000000BA  001F    268 CHECKSUM_BA=^XBA
000000BF  001F    269 CHECKSUM_BF=^XBF
000000C0  001F    270 CHECKSUM_C0=^XC0
000000C1  001F    271 CHECKSUM_C1=^XC1
000000C2  001F    272 CHECKSUM_C2=^XC2
000000C5  001F    273 CHECKSUM_C5=^XC5
000000C9  001F    274 CHECKSUM_C9=^XC9
000000DE  001F    275 CHECKSUM_DE=^XDE
000000DF  001F    276 CHECKSUM_DF=^XDF
000000E0  001F    277 CHECKSUM_E0=^XE0
000000E5  001F    278 CHECKSUM_E5=^XE5
000000E8  001F    279 CHECKSUM_E8=^XE8
000000F4  001F    280 CHECKSUM_F4=^XF4
000000F7  001F    281 CHECKSUM_F7=^XF7
000000FA  001F    282 CHECKSUM_FA=^XFA
000000FC  001F    283 CHECKSUM_FC=^XFC
          001F    284
          001F    285 ; Block number list
          001F    286
00000000  001F    287 BLOCK_00=^X00
00000001  001F    288 BLOCK_01=^X01
00000002  001F    289 BLOCK_02=^X02
00000004  001F    290 BLOCK_04=^X04
00000005  001F    291 BLOCK_05=^X05
00000006  001F    292 BLOCK_06=^X06
0000000A  001F    293 BLOCK_0A=^X0A
0000000C  001F    294 BLOCK_0C=^X0C
0000000D  001F    295 BLOCK_0D=^X0D
0000000E  001F    296 BLOCK_0E=^X0E
0000000F  001F    297 BLOCK_0F=^X0F
00000015  001F    298 BLOCK_15=^X15
0000005A  001F    299 BLOCK_5A=^X5A
00000080  001F    300 BLOCK_80=^X80
0000008F  001F    301 BLOCK_8F=^X8F
          001F    302
          001F    303 ; Byte counts
          001F    304
00000000  001F    305 BYTE_COUNT_00=^X00
00000001  001F    306 BYTE_COUNT_01=^X01
00000002  001F    307 BYTE_COUNT_02=^X02
00000004  001F    308 BYTE_COUNT_04=^X04
0000000A  001F    309 BYTE_COUNT_0A=^X0A
```

```
0000001A  001F    310 BYTE_COUNT_1A=^X1A
00000012  001F    311 BYTE_COUNT_12=^X12
00000064  001F    312 BYTE_COUNT_64=^X64
00000074  001F    313 BYTE_COUNT_74=^X74
00000080  001F    314 BYTE_COUNT_80=^X80
000000F4  001F    315 BYTE_COUNT_F4=^XF4
          001F    316
          001F    317 ; data
          001F    318
00000000  001F    319 DATA_0000=^X0000
00000001  001F    320 DATA_0001=^X0001
00000004  001F    321 DATA_0004=^X0004
0000000F  001F    322 DATA_000F=^X000F
00000015  001F    323 DATA_0015=^X0015
00000080  001F    324 DATA_0080=^X0080
000000AA  001F    325 DATA_00AA=^X00AA
000000FF  001F    326 DATA_00FF=^X00FF
00000100  001F    327 DATA_0100=^X0100
00000101  001F    328 DATA_0101=^X0101
00000115  001F    329 DATA_0115=^X0115
00000180  001F    330 DATA_0180=^X0180
00000204  001F    331 DATA_0204=^X0204
00000206  001F    332 DATA_0206=^X0206
0000025A  001F    333 DATA_025A=^X025A
0000065A  001F    334 DATA_065A=^X065A
00000400  001F    335 DATA_0400=^X0400
00000604  001F    336 DATA_0604=^X0604
00000605  001F    337 DATA_0605=^X0605
00000606  001F    338 DATA_0606=^X0606
00000800  001F    339 DATA_0800=^X0800
00000960  001F    340 DATA_0960=^X0960
00001400  001F    341 DATA_1400=^X1400
00001D14  001F    342 DATA_1D14=^X1D14
00004A77  001F    343 DATA_4A77=^X4A77
00007D09  001F    344 DATA_7D09=^X7D09
0000774A  001F    345 DATA_774A=^X774A
0000FFFF  001F    346 DATA_FFFF=^XFFFF
          001F    347
          001F    348
```

```
                    001F   350 .SBTTL  COMMAND PACKET TABLE
                    001F   351
                    001F   352 ; Single byte commands
                    001F   353
                    001F   354 B_INIT_DRIVE:                           ; initialize drive
                04  001F   355         .BYTE INIT_FLAG
                04  0020   356         .BYTE INIT_FLAG
                    0021   357
                    0021   358 B_ZEROS:                                ; zero bytes for init sequence
              0000  0021   359         .WORD   DATA_0000
                    0023   360
                    0023   361 B_CONTINUE:                             ; continue
                10  0023   362         .BYTE CONTINUE_FLAG
                    0024   363
                    0024   364 ; Multiple byte commands
                    0024   365
                    0024   366
                    0024   367 ; commands for RSP
                    0024   368
                    0024   369 B_INIT_CMD:
                02  0024   370         .BYTE   COMMAND_FLAG
                0A  0025   371         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
                01  0026   372         .BYTE   INITIALIZE              ; command
                00  0027   373         .BYTE   MODIFIER_00
                00  0028   374         .BYTE   UNIT_0
                00  0029   375         .BYTE   SWITCH_00
                00  002A   376         .BYTE   SEQUENCE_L
                00  002B   377         .BYTE   SEQUENCE_H
                00  002C   378         .BYTE   BYTE_COUNT_00           ; low (data)
                00  002D   379         .BYTE   BYTE_COUNT_00           ; high (data)
                00  002E   380         .BYTE   BLOCK_00                ; low
                00  002F   381         .BYTE   BLOCK_00                ; high
                03  0030   382         .BYTE   CHECKSUM_03             ; low
                0A  0031   383         .BYTE   CHECKSUM_0A             ; high
                    0032   384
                    0032   385
                    0032   386 B_NOP_CMD:
                02  0032   387         .BYTE   COMMAND_FLAG
                0A  0033   388         .BYTE   BYTE_COUNT_0A
                00  0034   389         .BYTE   NO_OPERATION            ; command
                00  0035   390         .BYTE   MODIFIER_00
                00  0036   391         .BYTE   UNIT_0
                00  0037   392         .BYTE   SWITCH_00
                00  0038   393         .BYTE   SEQUENCE_L
                00  0039   394         .BYTE   SEQUENCE_H
                00  003A   395         .BYTE   BYTE_COUNT_00           ; low (data)
                00  003B   396         .BYTE   BYTE_COUNT_00           ; high (data)
                00  003C   397         .BYTE   BLOCK_00                ; low
                00  003D   398         .BYTE   BLOCK_00                ; high
                02  003E   399         .BYTE   CHECKSUM_02             ; low
                0A  003F   400         .BYTE   CHECKSUM_0A             ; high
                    0040   401
                    0040   402
                    0040   403 B_DIAGNOSE_CMD:
                02  0040   404         .BYTE   COMMAND_FLAG
```

```
 0A  0041    405          .BYTE    BYTE_COUNT_0A
 07  0042    406          .BYTE    DIAGNOSE                   ; command
 00  0043    407          .BYTE    MODIFIER_00
 00  0044    408          .BYTE    UNIT_0
 00  0045    409          .BYTE    SWITCH_00
 00  0046    410          .BYTE    SEQUENCE_L
 00  0047    411          .BYTE    SEQUENCE_H
 00  0048    412          .BYTE    BYTE_COUNT_00              ; low (data)
 00  0049    413          .BYTE    BYTE_COUNT_00              ; high (data)
 00  004A    414          .BYTE    BLOCK_00                   ; low
 00  004B    415          .BYTE    BLOCK_00                   ; high
 09  004C    416          .BYTE    CHECKSUM_09                ; low
 0A  004D    417          .BYTE    CHECKSUM_0A                ; high
     004E    418
     004E    420 B_READ_1_CMD:
 02  004E    421            .BYTE    COMMAND_FLAG
 0A  004F    422            .BYTE    BYTE_COUNT_0A           ; byte count for command packet
 02  0050    423            .BYTE    READ                    ; command
 00  0051    424            .BYTE    MODIFIER_00
 00  0052    425            .BYTE    UNIT_0
 00  0053    426            .BYTE    SWITCH_00
 00  0054    427            .BYTE    SEQUENCE_L
 00  0055    428            .BYTE    SEQUENCE_H
 80  0056    429            .BYTE    BYTE_COUNT_80           ; low (data)
 00  0057    430            .BYTE    BYTE_COUNT_00           ; high (data)
 06  0058    431            .BYTE    BLOCK_06                ; low
 00  0059    432            .BYTE    BLOCK_00                ; high
 8A  005A    433            .BYTE    CHECKSUM_8A             ; low
 0A  005B    434            .BYTE    CHECKSUM_0A             ; high
     005C    435
     005C    436
     005C    437 B_WRITE_1_CMD:
 02  005C    438            .BYTE    COMMAND_FLAG
 0A  005D    439            .BYTE    BYTE_COUNT_0A           ; byte count for command packet
 03  005E    440            .BYTE    WRITE                   ; command
 00  005F    441            .BYTE    MODIFIER_00
 00  0060    442            .BYTE    UNIT_0
 00  0061    443            .BYTE    SWITCH_00
 00  0062    444            .BYTE    SEQUENCE_L
 00  0063    445            .BYTE    SEQUENCE_H
 12  0064    446            .BYTE    BYTE_COUNT_12           ; low (data)
 00  0065    447            .BYTE    BYTE_COUNT_00           ; high (data)
 06  0066    448            .BYTE    BLOCK_06                ; low
 00  0067    449            .BYTE    BLOCK_00                ; high
 1D  0068    450            .BYTE    CHECKSUM_1D             ; low
 0A  0069    451            .BYTE    CHECKSUM_0A             ; high
     006A    452
     006A    453
     006A    454 B_WRITE_2_CMD:
 02  006A    455            .BYTE    COMMAND_FLAG
 0A  006B    456            .BYTE    BYTE_COUNT_0A           ; byte count for command packet
 03  006C    457            .BYTE    WRITE                   ; command
 00  006D    458            .BYTE    MODIFIER_00
 00  006E    459            .BYTE    UNIT_0
 00  006F    460            .BYTE    SWITCH_00
 00  0070    461            .BYTE    SEQUENCE_L
 00  0071    462            .BYTE    SEQUENCE_H
 00  0072    463            .BYTE    BYTE_COUNT_00           ; low (data)
 02  0073    464            .BYTE    BYTE_COUNT_02           ; high (data)
 0A  0074    465            .BYTE    BLOCK_0A                ; low
 00  0075    466            .BYTE    BLOCK_00                ; high
 0F  0076    467            .BYTE    CHECKSUM_0F             ; low
 0C  0077    468            .BYTE    CHECKSUM_0C             ; high
```

```
      0078   470 B_READ_2_CMD:
 02   0078   471           .BYTE    COMMAND_FLAG
 0A   0079   472           .BYTE    BYTE_COUNT_0A            ; byte count for command packet
 02   007A   473           .BYTE    READ                     ; command
 00   007B   474           .BYTE    MODIFIER_00
 00   007C   475           .BYTE    UNIT_0
 00   007D   476           .BYTE    SWITCH_00
 00   007E   477           .BYTE    SEQUENCE_L
 00   007F   478           .BYTE    SEQUENCE_H
 00   0080   479           .BYTE    BYTE_COUNT_00            ; low (data)
 02   0081   480           .BYTE    BYTE_COUNT_02            ; high (data)
 0A   0082   481           .BYTE    BLOCK_0A                 ; low
 00   0083   482           .BYTE    BLOCK_00                 ; high
 0E   0084   483           .BYTE    CHECKSUM_0E              ; low
 0C   0085   484           .BYTE    CHECKSUM_0C              ; high
      0086   485
      0086   486
      0086   487 B_WRITE_3_CMD:
 02   0086   488           .BYTE    COMMAND_FLAG
 0A   0087   489           .BYTE    BYTE_COUNT_0A            ; byte count for command packet
 03   0088   490           .BYTE    WRITE                    ; command
 80   0089   491           .BYTE    MODIFIER_80
 00   008A   492           .BYTE    UNIT_0
 00   008B   493           .BYTE    SWITCH_00
 00   008C   494           .BYTE    SEQUENCE_L
 00   008D   495           .BYTE    SEQUENCE_H
 80   008E   496           .BYTE    BYTE_COUNT_80            ; low (data)
 00   008F   497           .BYTE    BYTE_COUNT_00            ; high (data)
 0E   0090   498           .BYTE    BLOCK_0E                 ; low
 00   0091   499           .BYTE    BLOCK_00                 ; high
 93   0092   500           .BYTE    CHECKSUM_93              ; low
 8A   0093   501           .BYTE    CHECKSUM_8A              ; high
      0094   502
      0094   503
      0094   504 B_READ_3_CMD:
 02   0094   505           .BYTE    COMMAND_FLAG
 0A   0095   506           .BYTE    BYTE_COUNT_0A            ; byte count for command packet
 02   0096   507           .BYTE    READ                     ; command
 80   0097   508           .BYTE    MODIFIER_80
 00   0098   509           .BYTE    UNIT_0
 00   0099   510           .BYTE    SWITCH_00
 00   009A   511           .BYTE    SEQUENCE_L
 00   009B   512           .BYTE    SEQUENCE_H
 80   009C   513           .BYTE    BYTE_COUNT_80            ; low (data)
 00   009D   514           .BYTE    BYTE_COUNT_00            ; high (data)
 0E   009E   515           .BYTE    BLOCK_0E                 ; low
 00   009F   516           .BYTE    BLOCK_00                 ; high
 92   00A0   517           .BYTE    CHECKSUM_92              ; low
 8A   00A1   518           .BYTE    CHECKSUM_8A              ; high
```

```
       00A2    520 B_WRITE_4_CMD:
  02   00A2    521          .BYTE   COMMAND_FLAG
  0A   00A3    522          .BYTE   BYTE_COUNT_0A              ; byte count for command packet
  03   00A4    523          .BYTE   WRITE                      ; command
  01   00A5    524          .BYTE   MODIFIER_01
  00   00A6    525          .BYTE   UNIT_0
  00   00A7    526          .BYTE   SWITCH_00
  00   00A8    527          .BYTE   SEQUENCE_L
  00   00A9    528          .BYTE   SEQUENCE_H
  00   00AA    529          .BYTE   BYTE_COUNT_00              ; low (data)
  02   00AB    530          .BYTE   BYTE_COUNT_02              ; high (data)
  0C   00AC    531          .BYTE   BLOCK_0C                   ; low
  00   00AD    532          .BYTE   BLOCK_00                   ; high
  11   00AE    533          .BYTE   CHECKSUM_11                ; low
  0D   00AF    534          .BYTE   CHECKSUM_0D                ; high
       00B0    535
       00B0    536
       00B0    537 B_READ_4_CMD:
  02   00B0    538          .BYTE   COMMAND_FLAG
  0A   00B1    539          .BYTE   BYTE_COUNT_0A              ; byte count for command packet
  02   00B2    540          .BYTE   READ                       ; command
  01   00B3    541          .BYTE   MODIFIER_01
  00   00B4    542          .BYTE   UNIT_0
  00   00B5    543          .BYTE   SWITCH_00
  00   00B6    544          .BYTE   SEQUENCE_L
  00   00B7    545          .BYTE   SEQUENCE_H
  00   00B8    546          .BYTE   BYTE_COUNT_00              ; low (data)
  02   00B9    547          .BYTE   BYTE_COUNT_02              ; high (data)
  0C   00BA    548          .BYTE   BLOCK_0C                   ; low
  00   00BB    549          .BYTE   BLOCK_00                   ; high
  10   00BC    550          .BYTE   CHECKSUM_10                ; low
  0D   00BD    551          .BYTE   CHECKSUM_0D                ; high
       00BE    552
       00BE    553
       00BE    554 B_WRITE_5_CMD:
  02   00BE    555          .BYTE   COMMAND_FLAG
  0A   00BF    556          .BYTE   BYTE_COUNT_0A              ; byte count for command packet
  03   00C0    557          .BYTE   WRITE                      ; command
  00   00C1    558          .BYTE   MODIFIER_00
  00   00C2    559          .BYTE   UNIT_0
  00   00C3    560          .BYTE   SWITCH_00
  00   00C4    561          .BYTE   SEQUENCE_L
  00   00C5    562          .BYTE   SEQUENCE_H
  00   00C6    563          .BYTE   BYTE_COUNT_00              ; low (data)
  02   00C7    564          .BYTE   BYTE_COUNT_02              ; high (data)
  0F   00C8    565          .BYTE   BLOCK_0F                   ; low
  00   00C9    566          .BYTE   BLOCK_00                   ; high
  14   00CA    567          .BYTE   CHECKSUM_14                ; low
  0C   00CB    568          .BYTE   CHECKSUM_0C                ; high
       00CC    569
```

```
    00CC    571 B_WRITE_6_CMD:
02  00CC    572         .BYTE   COMMAND_FLAG
0A  00CD    573         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
03  00CE    574         .BYTE   WRITE                   ; command
00  00CF    575         .BYTE   MODIFIER_00
00  00D0    576         .BYTE   UNIT_0
00  00D1    577         .BYTE   SWITCH_00
00  00D2    578         .BYTE   SEQUENCE_L
00  00D3    579         .BYTE   SEQUENCE_H
00  00D4    580         .BYTE   BYTE_COUNT_00           ; low (data)
02  00D5    581         .BYTE   BYTE_COUNT_02           ; high (data)
8F  00D6    582         .BYTE   BLOCK_8F                ; low
00  00D7    583         .BYTE   BLOCK_00                ; high
94  00D8    584         .BYTE   CHECKSUM_94             ; low
0C  00D9    585         .BYTE   CHECKSUM_0C             ; high
    00DA    586
    00DA    587
    00DA    588 B_WRITE_7_CMD:
02  00DA    589         .BYTE   COMMAND_FLAG
0A  00DB    590         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
03  00DC    591         .BYTE   WRITE                   ; command
00  00DD    592         .BYTE   MODIFIER_00
00  00DE    593         .BYTE   UNIT_0
00  00DF    594         .BYTE   SWITCH_00
00  00E0    595         .BYTE   SEQUENCE_L
00  00E1    596         .BYTE   SEQUENCE_H
01  00E2    597         .BYTE   BYTE_COUNT_01           ; low (data)
00  00E3    598         .BYTE   BYTE_COUNT_00           ; high (data)
0F  00E4    599         .BYTE   BLOCK_0F                ; low
00  00E5    600         .BYTE   BLOCK_00                ; high
15  00E6    601         .BYTE   CHECKSUM_15             ; low
0A  00E7    602         .BYTE   CHECKSUM_0A             ; high
    00E8    603
    00E8    604
    00E8    605 B_READ_5_CMD:
02  00E8    606         .BYTE   COMMAND_FLAG
0A  00E9    607         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
02  00EA    608         .BYTE   READ                    ; command
00  00EB    609         .BYTE   MODIFIER_00
00  00EC    610         .BYTE   UNIT_0
00  00ED    611         .BYTE   SWITCH_00
00  00EE    612         .BYTE   SEQUENCE_L
00  00EF    613         .BYTE   SEQUENCE_H
00  00F0    614         .BYTE   BYTE_COUNT_00           ; low (data)
02  00F1    615         .BYTE   BYTE_COUNT_02           ; high (data)
0F  00F2    616         .BYTE   BLOCK_0F                ; low
00  00F3    617         .BYTE   BLOCK_00                ; high
13  00F4    618         .BYTE   CHECKSUM_13             ; low
0C  00F5    619         .BYTE   CHECKSUM_0C             ; high
    00F6    620
```

```
        00F6    622 B_READ_6_CMD:
   02   00F6    623         .BYTE   COMMAND_FLAG
   0A   00F7    624         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
   02   00F8    625         .BYTE   READ                    ; command
   00   00F9    626         .BYTE   MODIFIER_00
   00   00FA    627         .BYTE   UNIT_0
   00   00FB    628         .BYTE   SWITCH_00
   00   00FC    629         .BYTE   SEQUENCE_L
   00   00FD    630         .BYTE   SEQUENCE_H
   00   00FE    631         .BYTE   BYTE_COUNT_00           ; low (data)
   02   00FF    632         .BYTE   BYTE_COUNT_02           ; high (data)
   8F   0100    633         .BYTE   BLOCK_8F                ; low
   00   0101    634         .BYTE   BLOCK_00                ; high
   93   0102    635         .BYTE   CHECKSUM_93             ; low
   0C   0103    636         .BYTE   CHECKSUM_0C             ; high
        0104    637
        0104    638
        0104    639 B_WRITE_8_CMD:
   02   0104    640         .BYTE   COMMAND_FLAG
   0A   0105    641         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
   03   0106    642         .BYTE   WRITE                   ; command
   80   0107    643         .BYTE   MODIFIER_80
   00   0108    644         .BYTE   UNIT_0
   00   0109    645         .BYTE   SWITCH_00
   00   010A    646         .BYTE   SEQUENCE_L
   00   010B    647         .BYTE   SEQUENCE_H
   80   010C    648         .BYTE   BYTE_COUNT_80           ; low (data)
   00   010D    649         .BYTE   BYTE_COUNT_00           ; high (data)
   0D   010E    650         .BYTE   BLOCK_0D                ; low
   00   010F    651         .BYTE   BLOCK_00                ; high
   92   0110    652         .BYTE   CHECKSUM_92             ; low
   8A   0111    653         .BYTE   CHECKSUM_8A             ; high
        0112    654
        0112    655
        0112    656 B_WRITE_9_CMD:
   02   0112    657         .BYTE   COMMAND_FLAG
   0A   0113    658         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
   03   0114    659         .BYTE   WRITE                   ; command
   80   0115    660         .BYTE   MODIFIER_80
   00   0116    661         .BYTE   UNIT_0
   00   0117    662         .BYTE   SWITCH_00
   00   0118    663         .BYTE   SEQUENCE_L
   00   0119    664         .BYTE   SEQUENCE_H
   80   011A    665         .BYTE   BYTE_COUNT_80           ; low (data)
   00   011B    666         .BYTE   BYTE_COUNT_00           ; high (data)
   0D   011C    667         .BYTE   BLOCK_0D                ; low
   02   011D    668         .BYTE   BLOCK_02                ; high
   92   011E    669         .BYTE   CHECKSUM_92             ; low
   8C   011F    670         .BYTE   CHECKSUM_8C             ; high
        0120    671
        0120      673 B_WRITE_10_CMD:
  02    0120      674          .BYTE    COMMAND_FLAG
  0A    0121      675          .BYTE    BYTE_COUNT_0A              ; byte count for command packet
  03    0122      676          .BYTE    WRITE                      ; command
  80    0123      677          .BYTE    MODIFIER_80
  00    0124      678          .BYTE    UNIT_0
  00    0125      679          .BYTE    SWITCH_00
  00    0126      680          .BYTE    SEQUENCE_L
  00    0127      681          .BYTE    SEQUENCE_H
  01    0128      682          .BYTE    BYTE_COUNT_01              ; low (data)
  00    0129      683          .BYTE    BYTE_COUNT_00              ; high (data)
  0D    012A      684          .BYTE    BLOCK_0D                   ; low
  00    012B      685          .BYTE    BLOCK_00                   ; high
  13    012C      686          .BYTE    CHECKSUM_13                ; low
  8A    012D      687          .BYTE    CHECKSUM_8A                ; high
        012E      688
        012E      689
        012E      690 B_READ_7_CMD:
  02    012E      691          .BYTE    COMMAND_FLAG
  0A    012F      692          .BYTE    BYTE_COUNT_0A              ; byte count for command packet
  02    0130      693          .BYTE    READ                       ; command
  80    0131      694          .BYTE    MODIFIER_80
  00    0132      695          .BYTE    UNIT_0
  00    0133      696          .BYTE    SWITCH_00
  00    0134      697          .BYTE    SEQUENCE_L
  00    0135      698          .BYTE    SEQUENCE_H
  80    0136      699          .BYTE    BYTE_COUNT_80              ; low (data)
  00    0137      700          .BYTE    BYTE_COUNT_00              ; high (data)
  0D    0138      701          .BYTE    BLOCK_0D                   ; low
  00    0139      702          .BYTE    BLOCK_00                   ; high
  91    013A      703          .BYTE    CHECKSUM_91                ; low
  8A    013B      704          .BYTE    CHECKSUM_8A                ; high
        013C      705
        013C      706
        013C      707 B_READ_8_CMD:
  02    013C      708          .BYTE    COMMAND_FLAG
  0A    013D      709          .BYTE    BYTE_COUNT_0A              ; byte count for command packet
  02    013E      710          .BYTE    READ                       ; command
  80    013F      711          .BYTE    MODIFIER_80
  00    0140      712          .BYTE    UNIT_0
  00    0141      713          .BYTE    SWITCH_00
  00    0142      714          .BYTE    SEQUENCE_L
  00    0143      715          .BYTE    SEQUENCE_H
  80    0144      716          .BYTE    BYTE_COUNT_80              ; low (data)
  00    0145      717          .BYTE    BYTE_COUNT_00              ; high (data)
  0D    0146      718          .BYTE    BLOCK_0D                   ; low
  02    0147      719          .BYTE    BLOCK_02                   ; high
  91    0148      720          .BYTE    CHECKSUM_91                ; low
  8C    0149      721          .BYTE    CHECKSUM_8C                ; high
        014A      722
```

```
       014A    724 B_WRITE_11_CMD:
   02  014A    725          .BYTE   COMMAND_FLAG
   0A  014B    726          .BYTE   BYTE_COUNT_0A           ; byte count for command packet
   03  014C    727          .BYTE   WRITE                   ; command
   00  014D    728          .BYTE   MODIFIER_00
   00  014E    729          .BYTE   UNIT_0
   00  014F    730          .BYTE   SWITCH_00
   00  0150    731          .BYTE   SEQUENCE_L
   00  0151    732          .BYTE   SEQUENCE_H
   00  0152    733          .BYTE   BYTE_COUNT_00           ; low (data)
   04  0153    734          .BYTE   BYTE_COUNT_04           ; high (data)
   00  0154    735          .BYTE   BLOCK_00                ; low
   00  0155    736          .BYTE   BLOCK_00                ; high
   05  0156    737          .BYTE   CHECKSUM_05             ; low
   0E  0157    738          .BYTE   CHECKSUM_0E             ; high
       0158    739
       0158    740
       0158    741 B_READ_9_CMD:
   02  0158    742          .BYTE   COMMAND_FLAG
   0A  0159    743          .BYTE   BYTE_COUNT_0A           ; byte count for command packet
   02  015A    744          .BYTE   READ                    ; command
   00  015B    745          .BYTE   MODIFIER_00
   00  015C    746          .BYTE   UNIT_0
   00  015D    747          .BYTE   SWITCH_00
   00  015E    748          .BYTE   SEQUENCE_L
   00  015F    749          .BYTE   SEQUENCE_H
   00  0160    750          .BYTE   BYTE_COUNT_00           ; low (data)
   04  0161    751          .BYTE   BYTE_COUNT_04           ; high (data)
   00  0162    752          .BYTE   BLOCK_00                ; low
   00  0163    753          .BYTE   BLOCK_00                ; high
   04  0164    754          .BYTE   CHECKSUM_04             ; low
   0E  0165    755          .BYTE   CHECKSUM_0E             ; high
       0166    756
       0166    757
       0166    758 B_WRITE_12_CMD:
   02  0166    759          .BYTE   COMMAND_FLAG
   0A  0167    760          .BYTE   BYTE_COUNT_0A           ; byte count for command packet
   03  0168    761          .BYTE   WRITE                   ; command
   00  0169    762          .BYTE   MODIFIER_00
   00  016A    763          .BYTE   UNIT_0
   00  016B    764          .BYTE   SWITCH_00
   00  016C    765          .BYTE   SEQUENCE_L
   00  016D    766          .BYTE   SEQUENCE_H
   00  016E    767          .BYTE   BYTE_COUNT_00           ; low (data)
   02  016F    768          .BYTE   BYTE_COUNT_02           ; high (data)
   80  0170    769          .BYTE   BLOCK_80                ; low
   01  0171    770          .BYTE   BLOCK_01                ; high
   85  0172    771          .BYTE   CHECKSUM_85             ; low
   0D  0173    772          .BYTE   CHECKSUM_0D             ; high
       0174    773
       0174    775 B_READ_10_CMD:
  02   0174    776          .BYTE   COMMAND_FLAG
  0A   0175    777          .BYTE   BYTE_COUNT_0A           ; byte count for command packet
  02   0176    778          .BYTE   READ                    ; command
  00   0177    779          .BYTE   MODIFIER_00
  00   0178    780          .BYTE   UNIT_0
  00   0179    781          .BYTE   SWITCH_00
  00   017A    782          .BYTE   SEQUENCE_L
  00   017B    783          .BYTE   SEQUENCE_H
  00   017C    784          .BYTE   BYTE_COUNT_00           ; low (data)
  02   017D    785          .BYTE   BYTE_COUNT_02           ; high (data)
  80   017E    786          .BYTE   BLOCK_80                ; low
  01   017F    787          .BYTE   BLOCK_01                ; high
  84   0180    788          .BYTE   CHECKSUM_84             ; low
  0D   0181    789          .BYTE   CHECKSUM_0D             ; high
       0182    790
       0182    791
       0182    792 B_WRITE_13_CMD:
  02   0182    793          .BYTE   COMMAND_FLAG
  0A   0183    794          .BYTE   BYTE_COUNT_0A           ; byte count for command packet
  03   0184    795          .BYTE   WRITE                   ; command
  00   0185    796          .BYTE   MODIFIER_00
  00   0186    797          .BYTE   UNIT_0
  00   0187    798          .BYTE   SWITCH_00
  00   0188    799          .BYTE   SEQUENCE_L
  00   0189    800          .BYTE   SEQUENCE_H
  00   018A    801          .BYTE   BYTE_COUNT_00           ; low (data)
  02   018B    802          .BYTE   BYTE_COUNT_02           ; high (data)
  00   018C    803          .BYTE   BLOCK_00                ; low
  01   018D    804          .BYTE   BLOCK_01                ; high
  05   018E    805          .BYTE   CHECKSUM_05             ; low
  0D   018F    806          .BYTE   CHECKSUM_0D             ; high
       0190    807
       0190    808
       0190    809 B_READ_11_CMD:
  02   0190    810          .BYTE   COMMAND_FLAG
  0A   0191    811          .BYTE   BYTE_COUNT_0A           ; byte count for command packet
  02   0192    812          .BYTE   READ                    ; command
  00   0193    813          .BYTE   MODIFIER_00
  00   0194    814          .BYTE   UNIT_0
  00   0195    815          .BYTE   SWITCH_00
  00   0196    816          .BYTE   SEQUENCE_L
  00   0197    817          .BYTE   SEQUENCE_H
  00   0198    818          .BYTE   BYTE_COUNT_00           ; low (data)
  02   0199    819          .BYTE   BYTE_COUNT_02           ; high (data)
  00   019A    820          .BYTE   BLOCK_00                ; low
  01   019B    821          .BYTE   BLOCK_01                ; high
  04   019C    822          .BYTE   CHECKSUM_04             ; low
  0D   019D    823          .BYTE   CHECKSUM_0D             ; high
       019E    824
```

```
             019E    826 B_WRITE_14_CMD:
         02  019E    827         .BYTE   COMMAND_FLAG
         0A  019F    828         .BYTE   BYTE_COUNT_0A                   ; byte count for command packet
         03  01A0    829         .BYTE   WRITE                           ; command
         00  01A1    830         .BYTE   MODIFIER_00
         00  01A2    831         .BYTE   UNIT_0
         00  01A3    832         .BYTE   SWITCH_00
         00  01A4    833         .BYTE   SEQUENCE_L
         00  01A5    834         .BYTE   SEQUENCE_H
         F4  01A6    835         .BYTE   BYTE_COUNT_F4                   ; low (data)
         01  01A7    836         .BYTE   BYTE_COUNT_01                   ; high (data)
         01  01A8    837         .BYTE   BLOCK_01                        ; low
         01  01A9    838         .BYTE   BLOCK_01                        ; high
         FA  01AA    839         .BYTE   CHECKSUM_FA                     ; low
         0C  01AB    840         .BYTE   CHECKSUM_0C                     ; high
             01AC    841
             01AC    842
             01AC    843 B_READ_12_CMD:
         02  01AC    844         .BYTE   COMMAND_FLAG
         0A  01AD    845         .BYTE   BYTE_COUNT_0A                   ; byte count for command packet
         02  01AE    846         .BYTE   READ                            ; command
         00  01AF    847         .BYTE   MODIFIER_00
         00  01B0    848         .BYTE   UNIT_0
         00  01B1    849         .BYTE   SWITCH_00
         00  01B2    850         .BYTE   SEQUENCE_L
         00  01B3    851         .BYTE   SEQUENCE_H
         00  01B4    852         .BYTE   BYTE_COUNT_00                   ; low (data)
         02  01B5    853         .BYTE   BYTE_COUNT_02                   ; high (data)
         01  01B6    854         .BYTE   BLOCK_01                        ; low
         01  01B7    855         .BYTE   BLOCK_01                        ; high
         05  01B8    856         .BYTE   CHECKSUM_05                     ; low
         0D  01B9    857         .BYTE   CHECKSUM_0D                     ; high
             01BA    858
             01BA    859
             01BA    860 B_WRITE_15_CMD:
         02  01BA    861         .BYTE   COMMAND_FLAG
         0A  01BB    862         .BYTE   BYTE_COUNT_0A                   ; byte count for command packet
         03  01BC    863         .BYTE   WRITE                           ; command
         00  01BD    864         .BYTE   MODIFIER_00
         00  01BE    865         .BYTE   UNIT_0
         00  01BF    866         .BYTE   SWITCH_00
         00  01C0    867         .BYTE   SEQUENCE_L
         00  01C1    868         .BYTE   SEQUENCE_H
         00  01C2    869         .BYTE   BYTE_COUNT_00                   ; low (data)
         02  01C3    870         .BYTE   BYTE_COUNT_02                   ; high (data)
         15  01C4    871         .BYTE   BLOCK_15                        ; low
         00  01C5    872         .BYTE   BLOCK_00                        ; high
         1A  01C6    873         .BYTE   CHECKSUM_1A                     ; low
         0C  01C7    874         .BYTE   CHECKSUM_0C                     ; high
             01C8    875
```

```
       01C8    877 B_READ_13_CMD:
   02  01C8    878         .BYTE   COMMAND_FLAG
   0A  01C9    879         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
   02  01CA    880         .BYTE   READ                    ; command
   00  01CB    881         .BYTE   MODIFIER_00
   00  01CC    882         .BYTE   UNIT_0
   00  01CD    883         .BYTE   SWITCH_00
   00  01CE    884         .BYTE   SEQUENCE_L
   00  01CF    885         .BYTE   SEQUENCE_H
   00  01D0    886         .BYTE   BYTE_COUNT_00           ; low (data)
   02  01D1    887         .BYTE   BYTE_COUNT_02           ; high (data)
   15  01D2    888         .BYTE   BLOCK_15                ; low
   00  01D3    889         .BYTE   BLOCK_00                ; high
   19  01D4    890         .BYTE   CHECKSUM_19             ; low
   0C  01D5    891         .BYTE   CHECKSUM_0C             ; high
       01D6    892
       01D6    893
       01D6    894 B_WRITE_16_CMD:
   02  01D6    895         .BYTE   COMMAND_FLAG
   0A  01D7    896         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
   03  01D8    897         .BYTE   WRITE                   ; command
   00  01D9    898         .BYTE   MODIFIER_00
   00  01DA    899         .BYTE   UNIT_0
   00  01DB    900         .BYTE   SWITCH_00
   00  01DC    901         .BYTE   SEQUENCE_L
   00  01DD    902         .BYTE   SEQUENCE_H
   00  01DE    903         .BYTE   BYTE_COUNT_00           ; low (data)
   02  01DF    904         .BYTE   BYTE_COUNT_02           ; high (data)
   15  01E0    905         .BYTE   BLOCK_15                ; low
   01  01E1    906         .BYTE   BLOCK_01                ; high
   1A  01E2    907         .BYTE   CHECKSUM_1A             ; low
   0D  01E3    908         .BYTE   CHECKSUM_0D             ; high
       01E4    909
       01E4    910
       01E4    911 B_READ_14_CMD:
   02  01E4    912         .BYTE   COMMAND_FLAG
   0A  01E5    913         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
   02  01E6    914         .BYTE   READ                    ; command
   00  01E7    915         .BYTE   MODIFIER_00
   00  01E8    916         .BYTE   UNIT_0
   00  01E9    917         .BYTE   SWITCH_00
   00  01EA    918         .BYTE   SEQUENCE_L
   00  01EB    919         .BYTE   SEQUENCE_H
   00  01EC    920         .BYTE   BYTE_COUNT_00           ; low (data)
   02  01ED    921         .BYTE   BYTE_COUNT_02           ; high (data)
   15  01EE    922         .BYTE   BLOCK_15                ; low
   01  01EF    923         .BYTE   BLOCK_01                ; high
   19  01F0    924         .BYTE   CHECKSUM_19             ; low
   0D  01F1    925         .BYTE   CHECKSUM_0D             ; high
```

```
        01F2   927 B_WRITE_17_CMD:
   02   01F2   928          .BYTE   COMMAND_FLAG
   0A   01F3   929          .BYTE   BYTE_COUNT_0A          ; byte count for command packet
   03   01F4   930          .BYTE   WRITE                  ; command
   00   01F5   931          .BYTE   MODIFIER_00
   00   01F6   932          .BYTE   UNIT_0
   00   01F7   933          .BYTE   SWITCH_00
   00   01F8   934          .BYTE   SEQUENCE_L
   00   01F9   935          .BYTE   SEQUENCE_H
   F4   01FA   936          .BYTE   BYTE_COUNT_F4          ; low (data)
   01   01FB   937          .BYTE   BYTE_COUNT_01          ; high (data)
   80   01FC   938          .BYTE   BLOCK_80               ; low
   00   01FD   939          .BYTE   BLOCK_00               ; high
   79   01FE   940          .BYTE   CHECKSUM_79            ; low
   0C   01FF   941          .BYTE   CHECKSUM_0C            ; high
        0200   942
        0200   943
        0200   944 B_READ_15_CMD:
   02   0200   945          .BYTE   COMMAND_FLAG
   0A   0201   946          .BYTE   BYTE_COUNT_0A          ; byte count for command packet
   02   0202   947          .BYTE   READ                   ; command
   00   0203   948          .BYTE   MODIFIER_00
   00   0204   949          .BYTE   UNIT_0
   00   0205   950          .BYTE   SWITCH_00
   00   0206   951          .BYTE   SEQUENCE_L
   00   0207   952          .BYTE   SEQUENCE_H
   00   0208   953          .BYTE   BYTE_COUNT_00          ; low (data)
   02   0209   954          .BYTE   BYTE_COUNT_02          ; high (data)
   80   020A   955          .BYTE   BLOCK_80               ; low
   00   020B   956          .BYTE   BLOCK_00               ; high
   84   020C   957          .BYTE   CHECKSUM_84            ; low
   0C   020D   958          .BYTE   CHECKSUM_0C            ; high
        020E   959
        020E   960
        020E   961 B_WRITE_18_CMD:
   02   020E   962          .BYTE   COMMAND_FLAG
   0A   020F   963          .BYTE   BYTE_COUNT_0A          ; byte count for command packet
   03   0210   964          .BYTE   WRITE                  ; command
   80   0211   965          .BYTE   MODIFIER_80
   00   0212   966          .BYTE   UNIT_0
   00   0213   967          .BYTE   SWITCH_00
   00   0214   968          .BYTE   SEQUENCE_L
   00   0215   969          .BYTE   SEQUENCE_H
   00   0216   970          .BYTE   BYTE_COUNT_00          ; low (data)
   01   0217   971          .BYTE   BYTE_COUNT_01          ; high (data)
   04   0218   972          .BYTE   BLOCK_04               ; low
   02   0219   973          .BYTE   BLOCK_02               ; high
   09   021A   974          .BYTE   CHECKSUM_09            ; low
   8D   021B   975          .BYTE   CHECKSUM_8D            ; high
```

```
      021C    977 B_READ_16_CMD:
  02  021C    978          .BYTE    COMMAND_FLAG
  0A  021D    979          .BYTE    BYTE_COUNT_0A            ; byte count for command packet
  02  021E    980          .BYTE    READ                     ; command
  80  021F    981          .BYTE    MODIFIER_80
  00  0220    982          .BYTE    UNIT_0
  00  0221    983          .BYTE    SWITCH_00
  00  0222    984          .BYTE    SEQUENCE_L
  00  0223    985          .BYTE    SEQUENCE_H
  00  0224    986          .BYTE    BYTE_COUNT_00            ; low (data)
  01  0225    987          .BYTE    BYTE_COUNT_01            ; high (data)
  04  0226    988          .BYTE    BLOCK_04                 ; low
  02  0227    989          .BYTE    BLOCK_02                 ; high
  08  0228    990          .BYTE    CHECKSUM_08              ; low
  8D  0229    991          .BYTE    CHECKSUM_8D              ; high
      022A    992
      022A    993
      022A    994 B_WRITE_19_CMD:
  02  022A    995          .BYTE    COMMAND_FLAG
  0A  022B    996          .BYTE    BYTE_COUNT_0A            ; b;te count for command packet
  03  022C    997          .BYTE    WRITE                    ; command
  80  022D    998          .BYTE    MODIFIER_80
  00  022E    999          .BYTE    UNIT_0
  00  022F   1000          .BYTE    SWITCH_00
  00  0230   1001          .BYTE    SEQUENCE_L
  00  0231   1002          .BYTE    SEQUENCE_H
  80  0232   1003          .BYTE    BYTE_COUNT_80            ; low (data)
  00  0233   1004          .BYTE    BYTE_COUNT_00            ; high (data)
  04  0234   1005          .BYTE    BLOCK_04                 ; low
  06  0235   1006          .BYTE    BLOCK_06                 ; high
  89  0236   1007          .BYTE    CHECKSUM_89              ; low
  90  0237   1008          .BYTE    CHECKSUM_90              ; high
      0238   1009
      0238   1010
      0238   1011 B_READ_17_CMD:
  02  0238   1012          .BYTE    COMMAND_FLAG
  0A  0239   1013          .BYTE    BYTE_COUNT_0A            ; byte count for command packet
  02  023A   1014          .BYTE    READ                     ; command
  80  023B   1015          .BYTE    MODIFIER_80
  00  023C   1016          .BYTE    UNIT_0
  00  023D   1017          .BYTE    SWITCH_00
  00  023E   1018          .BYTE    SEQUENCE_L
  00  023F   1019          .RYTE    SEQUENCE_H
  80  0240   1020          .BYTE    BYTE_COUNT_80            ; low (data)
  00  0241   1021          .BYTE    BYTE_COUNT_00            ; high (data)
  04  0242   1022          .BYTE    BLOCK_04                 ; low
  06  0243   1023          .BYTE    BLOCK_06                 ; high
  88  0244   1024          .BYTE    CHECKSUM_88              ; low
  90  0245   1025          .BYTE    CHECKSUM_90              ; high
```

```
     0246  1027 B_WRITE_20_CMD:
 02  0246  1028         .BYTE   COMMAND_FLAG
 0A  0247  1029         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
 03  0248  1030         .BYTE   WRITE                   ; command
 80  0249  1031         .BYTE   MODIFIER_80
 00  024A  1032         .BYTE   UNIT_0
 00  024B  1033         .BYTE   SWITCH_00
 00  024C  1034         .BYTE   SEQUENCE_L
 00  024D  1035         .BYTE   SEQUENCE_H
 80  024E  1036         .BYTE   BYTE_COUNT_80           ; low (data)
 00  024F  1037         .BYTE   BYTE_COUNT_00           ; high (data)
 06  0250  1038         .BYTE   BLOCK_06                ; low
 06  0251  1039         .BYTE   BLOCK_06                ; high
 8B  0252  1040         .BYTE   CHECKSUM_8B             ; low
 90  0253  1041         .BYTE   CHECKSUM_90             ; high
     0254  1042
     0254  1043
     0254  1044 B_READ_18_CMD:
 02  0254  1045         .BYTE   COMMAND_FLAG
 0A  0255  1046         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
 02  0256  1047         .BYTE   READ                    ; command
 80  0257  1048         .BYTE   MODIFIER_80
 00  0258  1049         .BYTE   UNIT_0
 00  0259  1050         .BYTE   SWITCH_00
 00  025A  1051         .BYTE   SEQUENCE_L
 00  025B  1052         .BYTE   SEQUENCE_H
 80  025C  1053         .BYTE   BYTE_COUNT_80           ; low (data)
 00  025D  1054         .BYTE   BYTE_COUNT_00           ; high (data)
 06  025E  1055         .BYTE   BLOCK_06                ; low
 06  025F  1056         .BYTE   BLOCK_06                ; high
 8A  0260  1057         .BYTE   CHECKSUM_8A             ; low
 90  0261  1058         .BYTE   CHECKSUM_90             ; high
     0262  1059
     0262  1060
     0262  1061 B_WRITE_21_CMD:
 02  0262  1062         .BYTE   COMMAND_FLAG
 0A  0263  1063         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
 03  0264  1064         .BYTE   WRITE                   ; command
 80  0265  1065         .BYTE   MODIFIER_80
 00  0266  1066         .BYTE   UNIT_0
 00  0267  1067         .BYTE   SWITCH_00
 00  0268  1068         .BYTE   SEQUENCE_L
 00  0269  1069         .BYTE   SEQUENCE_H
 64  026A  1070         .BYTE   BYTE_COUNT_64           ; low (data)
 00  026B  1071         .BYTE   BYTE_COUNT_00           ; high (data)
 05  026C  1072         .BYTE   BLOCK_05                ; low
 06  026D  1073         .BYTE   BLOCK_06                ; high
 6E  026E  1074         .BYTE   CHECKSUM_6E             ; low
 90  026F  1075         .BYTE   CHECKSUM_90             ; high
     0270  1076
```

```
    0270  1078 B_READ_19_CMD:
02  0270  1079          .BYTE   COMMAND_FLAG
0A  0271  1080          .BYTE   BYTE_COUNT_0A           ; byte count for command packet
02  0272  1081          .BYTE   READ                    ; command
80  0273  1082          .BYTE   MODIFIER_80
00  0274  1083          .BYTE   UNIT_0
00  0275  1084          .BYTE   SWITCH_00
00  0276  1085          .BYTE   SEQUENCE_L
00  0277  1086          .BYTE   SEQUENCE_H
80  0278  1087          .BYTE   BYTE_COUNT_80           ; low (data)
00  0279  1088          .BYTE   BYTE_COUNT_00           ; high (data)
05  027A  1089          .BYTE   BLOCK_05                ; low
06  027B  1090          .BYTE   BLOCK_06                ; high
89  027C  1091          .BYTE   CHECKSUM_89             ; low
90  027D  1092          .BYTE   CHECKSUM_90             ; high
    027E  1093
    027E  1094
    027E  1095 B_WRITE_22_CMD:
02  027E  1096          .BYTE   COMMAND_FLAG
0A  027F  1097          .BYTE   BYTE_COUNT_0A           ; byte count for command packet
03  0280  1098          .BYTE   WRITE                   ; command
80  0281  1099          .BYTE   MODIFIER_80
00  0282  1100          .BYTE   UNIT_0
00  0283  1101          .BYTE   SWITCH_00
00  0284  1102          .BYTE   SEQUENCE_L
00  0285  1103          .BYTE   SEQUENCE_H
80  0286  1104          .BYTE   BYTE_COUNT_80           ; low (data)
00  0287  1105          .BYTE   BYTE_COUNT_00           ; high (data)
5A  0288  1106          .BYTE   BLOCK_5A                ; low
02  0289  1107          .BYTE   BLOCK_02                ; high
DF  028A  1108          .BYTE   CHECKSUM_DF             ; low
8C  028B  1109          .BYTE   CHECKSUM_8C             ; high
    028C  1110
    028C  1111
    028C  1112 B_READ_20_CMD:
02  028C  1113          .BYTE   COMMAND_FLAG
0A  028D  1114          .BYTE   BYTE_COUNT_0A           ; byte count for command packet
02  028E  1115          .BYTE   READ                    ; command
80  028F  1116          .BYTE   MODIFIER_80
00  0290  1117          .BYTE   UNIT_0
00  0291  1118          .BYTE   SWITCH_00
00  0292  1119          .BYTE   SEQUENCE_L
00  0293  1120          .BYTE   SEQUENCE_H
80  0294  1121          .BYTE   BYTE_COUNT_80           ; low (data)
00  0295  1122          .BYTE   BYTE_COUNT_00           ; high (data)
5A  0296  1123          .BYTE   BLOCK_5A                ; low
02  0297  1124          .BYTE   BLOCK_02                ; high
DE  0298  1125          .BYTE   CHECKSUM_DE             ; low
8C  0299  1126          .BYTE   CHECKSUM_8C             ; high
```

```
      029A  1128 B_WRITE_23_CMD:
  02  029A  1129          .BYTE   COMMAND_FLAG
  0A  029B  1130          .BYTE   BYTE_COUNT_0A             ; byte count for command packet
  03  029C  1131          .BYTE   WRITE                     ; command
  80  029D  1132          .BYTE   MODIFIER_80
  00  029E  1133          .BYTE   UNIT_0
  00  029F  1134          .BYTE   SWITCH_00
  00  02A0  1135          .BYTE   SEQUENCE_L
  00  02A1  1136          .BYTE   SEQUENCE_H
  80  02A2  1137          .BYTE   BYTE_COUNT_80             ; low (data)
  00  02A3  1138          .BYTE   BYTE_COUNT_00             ; high (data)
  5A  02A4  1139          .BYTE   BLOCK_5A                  ; low
  06  02A5  1140          .BYTE   BLOCK_06                  ; high
  DF  02A6  1141          .BYTE   CHECKSUM_DF               ; low
  90  02A7  1142          .BYTE   CHECKSUM_90               ; high
      02A8  1143
      02A8  1144
      02A8  1145 B_READ_21_CMD:
  02  02A8  1146          .BYTE   COMMAND_FLAG
  0A  02A9  1147          .BYTE   BYTE_COUNT_0A             ; byte count for command packet
  02  02AA  1148          .BYTE   READ                      ; command
  80  02AB  1149          .BYTE   MODIFIER_80
  00  02AC  1150          .BYTE   UNIT_0
  00  02AD  1151          .BYTE   SWITCH_00
  00  02AE  1152          .BYTE   SEQUENCE_L
  00  02AF  1153          .BYTE   SEQUENCE_H
  80  02B0  1154          .BYTE   BYTE_COUNT_80             ; low (data)
  00  02B1  1155          .BYTE   BYTE_COUNT_00             ; high (data)
  5A  02B2  1156          .BYTE   BLOCK_5A                  ; low
  06  02B3  1157          .BYTE   BLOCK_06                  ; high
  DE  02B4  1158          .BYTE   CHECKSUM_DE               ; low
  90  02B5  1159          .BYTE   CHECKSUM_90               ; high
      02B6  1160
      02B6  1161
      02B6  1162 B_WRITE_24_CMD:
  02  02B6  1163          .BYTE   COMMAND_FLAG
  0A  02B7  1164          .BYTE   BYTE_COUNT_0A             ; byte count for command packet
  03  02B8  1165          .BYTE   WRITE                     ; command
  80  02B9  1166          .BYTE   MODIFIER_80
  00  02BA  1167          .BYTE   UNIT_0
  00  02BB  1168          .BYTE   SWITCH_00
  00  02BC  1169          .BYTE   SEQUENCE_L
  00  02BD  1170          .BYTE   SEQUENCE_H
  64  02BE  1171          .BYTE   BYTE_COUNT_64             ; low (data)
  00  02BF  1172          .BYTE   BYTE_COUNT_00             ; high (data)
  06  02C0  1173          .BYTE   BLOCK_06                  ; low
  02  02C1  1174          .BYTE   BLOCK_02                  ; high
  6F  02C2  1175          .BYTE   CHECKSUM_6F               ; low
  8C  02C3  1176          .BYTE   CHECKSUM_8C               ; high
```

```
     02C4  1178 B_READ_22_CMD:
02   02C4  1179          .BYTE    COMMAND_FLAG
0A   02C5  1180          .BYTE    BYTE_COUNT_0A              ; byte count for command packet
02   02C6  1181          .BYTE    READ                       ; command
80   02C7  1182          .BYTE    MODIFIER_80
00   02C8  1183          .BYTE    UNIT_0
00   02C9  1184          .BYTE    SWITCH_00
00   02CA  1185          .BYTE    SEQUENCE_L
00   02CB  1186          .BYTE    SEQUENCE_H
80   02CC  1187          .BYTE    BYTE_COUNT_80              ; low (data)
00   02CD  1188          .BYTE    BYTE_COUNT_00              ; high (data)
06   02CE  1189          .BYTE    BLOCK_06                   ; low
02   02CF  1190          .BYTE    BLOCK_02                   ; high
8A   02D0  1191          .BYTE    CHECKSUM_8A                ; low
8C   02D1  1192          .BYTE    CHECKSUM_8C                ; high
     02D2  1193
     02D2  1194
     02D2  1195 B_READ_23_CMD:
02   02D2  1196          .BYTE    COMMAND_FLAG
0A   02D3  1197          .BYTE    BYTE_COUNT_0A              ; byte count for command packet
02   02D4  1198          .BYTE    READ                       ; command
00   02D5  1199          .BYTE    MODIFIER_00
00   02D6  1200          .BYTE    UNIT_0
00   02D7  1201          .BYTE    SWITCH_00
00   02D8  1202          .BYTE    SEQUENCE_L
00   02D9  1203          .BYTE    SEQUENCE_H
00   02DA  1204          .BYTE    BYTE_COUNT_00              ; low (data)
02   02DB  1205          .BYTE    BYTE_COUNT_02              ; high (data)
00   02DC  1206          .BYTE    BLOCK_00                   ; low
00   02DD  1207          .BYTE    BLOCK_00                   ; high
04   02DE  1208          .BYTE    CHECKSUM_04                ; low
0C   02DF  1209          .BYTE    CHECKSUM_0C                ; high
     02E0  1210
     02E0  1211
     02E0  1212
     02E0  1213
     02E0  1214 B_READ_24_CMD:
02   02E0  1215          .BYTE    COMMAND_FLAG
0A   02E1  1216          .BYTE    BYTE_COUNT_0A              ; byte count for command packet
02   02E2  1217          .BYTE    READ                       ; command
00   02E3  1218          .BYTE    MODIFIER_00
00   02E4  1219          .BYTE    UNIT_0
00   02E5  1220          .BYTE    SWITCH_00
00   02E6  1221          .BYTE    SEQUENCE_L
00   02E7  1222          .BYTE    SEQUENCE_H
00   02E8  1223          .BYTE    BYTE_COUNT_00              ; low (data)
02   02E9  1224          .BYTE    BYTE_COUNT_02              ; high (data)
15   02EA  1225          .BYTE    BLOCK_15                   ; low
00   02EB  1226          .BYTE    BLOCK_00                   ; high
19   02EC  1227          .BYTE    CHECKSUM_19                ; low
0C   02ED  1228          .BYTE    CHECKSUM_0C                ; high
     02EE  1229
```

```
           02EE  1231 B_POS_1_CMD:
      02   02EE  1232          .BYTE   COMMAND_FLAG
      0A   02EF  1233          .BYTE   BYTE_COUNT_0A             ; byte count for command packet
      05   02F0  1234          .BYTE   POSITION                  ; command
      00   02F1  1235          .BYTE   MODIFIER_00
      00   02F2  1236          .BYTE   UNIT_0
      00   02F3  1237          .BYTE   SWITCH_00
      00   02F4  1238          .BYTE   SEQUENCE_L
      00   02F5  1239          .BYTE   SEQUENCE_H
      00   02F6  1240          .BYTE   BYTE_COUNT_00             ; low (data)
      00   02F7  1241          .BYTE   BYTE_COUNT_00             ; high (data)
      15   02F8  1242          .BYTE   BLOCK_15                  ; low
      00   02F9  1243          .BYTE   BLOCK_00                  ; high
      1C   02FA  1244          .BYTE   CHECKSUM_1C               ; low
      0A   02FB  1245          .BYTE   CHECKSUM_0A               ; high
           02FC  1246
           02FC  1247 ; commands for MRSP
           02FC  1248
           02FC  1249 MB_INIT_CMD:
      02   02FC  1250          .BYTE   COMMAND_FLAG
      0A   02FD  1251          .BYTE   BYTE_COUNT_0A             ; byte count for command packet
      01   02FE  1252          .BYTE   INITIALIZE                ; command
      00   02FF  1253          .BYTE   MODIFIER_00
      00   0300  1254          .BYTE   UNIT_0
      08   0301  1255          .BYTE   SWITCH_08
      00   0302  1256          .BYTE   SEQUENCE_L
      00   0303  1257          .BYTE   SEQUENCE_H
      00   0304  1258          .BYTE   BYTE_COUNT_00             ; low (data)
      00   0305  1259          .BYTE   BYTE_COUNT_00             ; high (data)
      00   0306  1260          .BYTE   BLOCK_00                  ; low
      00   0307  1261          .BYTE   BLOCK_00                  ; high
      03   0308  1262          .BYTE   CHECKSUM_03               ; low
      12   0309  1263          .BYTE   CHECKSUM_12               ; high
           030A  1264
           030A  1265
           030A  1266 MB_NOP_CMD:
      02   030A  1267          .BYTE   COMMAND_FLAG
      0A   030B  1268          .BYTE   BYTE_COUNT_0A
      00   030C  1269          .BYTE   NO_OPERATION              ; command
      00   030D  1270          .BYTE   MODIFIER_00
      00   030E  1271          .BYTE   UNIT_0
      08   030F  1272          .BYTE   SWITCH_08
      00   0310  1273          .BYTE   SEQUENCE_L
      00   0311  1274          .BYTE   SEQUENCE_H
      00   0312  1275          .BYTE   BYTE_COUNT_00             ; low (data)
      00   0313  1276          .BYTE   BYTE_COUNT_00             ; high (data)
      00   0314  1277          .BYTE   BLOCK_00                  ; low
      00   0315  1278          .BYTE   BLOCK_00                  ; high
      02   0316  1279          .BYTE   CHECKSUM_02               ; low
      12   0317  1280          .BYTE   CHECKSUM_12               ; high
           0318  1281
           0318  1282
           0318  1283 MB_DIAGNOSE_CMD:
      02   0318  1284          .BYTE   COMMAND_FLAG
      0A   0319  1285          .BYTE   BYTE_COUNT_0A
```

```
07  031A  1286         .BYTE   DIAGNOSE              ; command
00  031B  1287         .BYTE   MODIFIER_00
00  031C  1288         .BYTE   UNIT_0
08  031D  1289         .BYTE   SWITCH_08
00  031E  1290         .BYTE   SEQUENCE_L
00  031F  1291         .BYTE   SEQUENCE_H
00  0320  1292         .BYTE   BYTE_COUNT_00         ; low (data)
00  0321  1293         .BYTE   BYTE_COUNT_00         ; high (data)
00  0322  1294         .BYTE   BLOCK_00              ; low
00  0323  1295         .BYTE   BLOCK_00              ; high
09  0324  1296         .BYTE   CHECKSUM_09           ; low
12  0325  1297         .BYTE   CHECKSUM_12           ; high
    0326  1298
```

```
     0326  MB_READ_1_CMD:
 02  0326          .BYTE    COMMAND_FLAG
 0A  0327          .BYTE    BYTE_COUNT_0A                 ; byte count for command packet
 02  0328          .BYTE    READ                          ; command
 00  0329          .BYTE    MODIFIER_00
 00  032A          .BYTE    UNIT_0
 08  032B          .BYTE    SWITCH_08
 00  032C          .BYTE    SEQUENCE_L
 00  032D          .BYTE    SEQUENCE_H
 80  032E          .BYTE    BYTE_COUNT_80                 ; low (data)
 00  032F          .BYTE    BYTE_COUNT_00                 ; high (data)
 06  0330          .BYTE    BLOCK_06                      ; low
 00  0331          .BYTE    BLOCK_00                      ; high
 8A  0332          .BYTE    CHECKSUM_8A                   ; low
 12  0333          .BYTE    CHECKSUM_12                   ; high
     0334
     0334
     0334  MB_WRITE_1_CMD:
 02  0334          .BYTE    COMMAND_FLAG
 0A  0335          .BYTE    BYTE_COUNT_0A                 ; byte count for command packet
 03  0336          .BYTE    WRITE                         ; command
 00  0337          .BYTE    MODIFIER_00
 00  0338          .BYTE    UNIT_0
 08  0339          .BYTE    SWITCH_08
 00  033A          .BYTE    SEQUENCE_L
 00  033B          .BYTE    SEQUENCE_H
 12  033C          .BYTE    BYTE_COUNT_12                 ; low (data)
 00  033D          .BYTE    BYTE_COUNT_00                 ; high (data)
 06  033E          .BYTE    BLOCK_06                      ; low
 00  033F          .BYTE    BLOCK_00                      ; high
 1D  0340          .BYTE    CHECKSUM_1D                   ; low
 12  0341          .BYTE    CHECKSUM_12                   ; high
     0342
     0342
     0342  MB_WRITE_2_CMD:
 02  0342          .BYTE    COMMAND_FLAG
 0A  0343          .BYTE    BYTE_COUNT_0A                 ; byte count for command packet
 03  0344          .BYTE    WRITE                         ; command
 00  0345          .BYTE    MODIFIER_00
 00  0346          .BYTE    UNIT_0
 08  0347          .BYTE    SWITCH_08
 00  0348          .BYTE    SEQUENCE_L
 00  0349          .BYTE    SEQUENCE_H
 00  034A          .BYTE    BYTE_COUNT_00                 ; low (data)
 02  034B          .BYTE    BYTE_COUNT_02                 ; high (data)
 0A  034C          .BYTE    BLOCK_0A                      ; low
 00  034D          .BYTE    BLOCK_00                      ; high
 0F  034E          .BYTE    CHECKSUM_0F                   ; low
 14  034F          .BYTE    CHECKSUM_14                   ; high
```

```
    0350  1350 MB_READ_2_CMD:
02  0350  1351          .BYTE   COMMAND_FLAG
0A  0351  1352          .BYTE   BYTE_COUNT_0A           ; byte count for command packet
02  0352  1353          .BYTE   READ                    ; command
00  0353  1354          .BYTE   MODIFIER_00
00  0354  1355          .BYTE   UNIT_0
08  0355  1356          .BYTE   SWITCH_08
00  0356  1357          .BYTE   SEQUENCE_L
00  0357  1358          .BYTE   SEQUENCE_H
00  0358  1359          .BYTE   BYTE_COUNT_00           ; low (data)
02  0359  1360          .BYTE   BYTE_COUNT_02           ; high (data)
0A  035A  1361          .BYTE   BLOCK_0A                ; low
00  035B  1362          .BYTE   BLOCK_00                ; high
0E  035C  1363          .BYTE   CHECKSUM_0E             ; low
14  035D  1364          .BYTE   CHECKSUM_14             ; high
    035E  1365
    035E  1366
    035E  1367 MB_WRITE_3_CMD:
02  035E  1368          .BYTE   COMMAND_FLAG
0A  035F  1369          .BYTE   BYTE_COUNT_0A           ; byte count for command packet
03  0360  1370          .BYTE   WRITE                   ; command
80  0361  1371          .BYTE   MODIFIER_80
00  0362  1372          .BYTE   UNIT_0
08  0363  1373          .BYTE   SWITCH_08
00  0364  1374          .BYTE   SEQUENCE_L
00  0365  1375          .BYTE   SEQUENCE_H
80  0366  1376          .BYTE   BYTE_COUNT_80           ; low (data)
00  0367  1377          .BYTE   BYTE_COUNT_00           ; high (data)
0E  0368  1378          .BYTE   BLOCK_0E                ; low
00  0369  1379          .BYTE   BLOCK_00                ; high
93  036A  1380          .BYTE   CHECKSUM_93             ; low
92  036B  1381          .BYTE   CHECKSUM_92             ; high
    036C  1382
    036C  1383
    036C  1384 MB_READ_3_CMD:
02  036C  1385          .BYTE   COMMAND_FLAG
0A  036D  1386          .BYTE   BYTE_COUNT_0A           ; byte count for command packet
02  036E  1387          .BYTE   READ                    ; command
80  036F  1388          .BYTE   MODIFIER_80
00  0370  1389          .BYTE   UNIT_0
08  0371  1390          .BYTE   SWITCH_08
00  0372  1391          .BYTE   SEQUENCE_L
00  0373  1392          .BYTE   SEQUENCE_H
80  0374  1393          .BYTE   BYTE_COUNT_80           ; low (data)
00  0375  1394          .BYTE   BYTE_COUNT_00           ; high (data)
0E  0376  1395          .BYTE   BLOCK_0E                ; low
00  0377  1396          .BYTE   BLOCK_00                ; high
92  0378  1397          .BYTE   CHECKSUM_92             ; low
92  0379  1398          .BYTE   CHECKSUM_92             ; high
```

```
      037A  1400 MB_WRITE_4_CMD:
  02  037A  1401          .BYTE   COMMAND_FLAG
  0A  037B  1402          .BYTE   BYTE_COUNT_0A              ; byte count for command packet
  03  037C  1403          .BYTE   WRITE                      ; command
  01  037D  1404          .BYTE   MODIFIER_01
  00  037E  1405          .BYTE   UNIT_0
  08  037F  1406          .BYTE   SWITCH_08
  00  0380  1407          .BYTE   SEQUENCE_L
  00  0381  1408          .BYTE   SEQUENCE_H
  00  0382  1409          .BYTE   BYTE_COUNT_00              ; low (data)
  02  0383  1410          .BYTE   BYTE_COUNT_02              ; high (data)
  0C  0384  1411          .BYTE   BLOCK_0C                   ; low
  00  0385  1412          .BYTE   BLOCK_00                   ; high
  11  0386  1413          .BYTE   CHECKSUM_11                ; low
  15  0387  1414          .BYTE   CHECKSUM_15                ; high
      0388  1415
      0388  1416
      0388  1417 MB_READ_4_CMD:
  02  0388  1418          .BYTE   COMMAND_FLAG
  0A  0389  1419          .BYTE   BYTE_COUNT_0A              ; byte count for command packet
  02  038A  1420          .BYTE   READ                       ; command
  01  038B  1421          .BYTE   MODIFIER_01
  00  038C  1422          .BYTE   UNIT_0
  08  038D  1423          .BYTE   SWITCH_08
  00  038E  1424          .BYTE   SEQUENCE_L
  00  038F  1425          .BYTE   SEQUENCE_H
  00  0390  1426          .BYTE   BYTE_COUNT_00              ; low (data)
  02  0391  1427          .BYTE   BYTE_COUNT_02              ; high (data)
  0C  0392  1428          .BYTE   BLOCK_0C                   ; low
  00  0393  1429          .BYTE   BLOCK_00                   ; high
  10  0394  1430          .BYTE   CHECKSUM_10                ; low
  15  0395  1431          .BYTE   CHECKSUM_15                ; high
      0396  1432
      0396  1433
      0396  1434 MB_WRITE_5_CMD:
  02  0396  1435          .BYTE   COMMAND_FLAG
  0A  0397  1436          .BYTE   BYTE_COUNT_0A              ; byte count for command packet
  03  0398  1437          .BYTE   WRITE                      ; command
  00  0399  1438          .BYTE   MODIFIER_00
  00  039A  1439          .BYTE   UNIT_0
  08  039B  1440          .BYTE   SWITCH_08
  00  039C  1441          .BYTE   SEQUENCE_L
  00  039D  1442          .BYTE   SEQUENCE_H
  00  039E  1443          .BYTE   BYTE_COUNT_00              ; low (data)
  02  039F  1444          .BYTE   BYTE_COUNT_02              ; high (data)
  0F  03A0  1445          .BYTE   BLOCK_0F                   ; low
  00  03A1  1446          .BYTE   BLOCK_00                   ; high
  14  03A2  1447          .BYTE   CHECKSUM_14                ; low
  14  03A3  1448          .BYTE   CHECKSUM_14                ; high
      03A4  1449
```

```
    03A4  1451 MB_WRITE_6_CMD:
02  03A4  1452          .BYTE   COMMAND_FLAG
0A  03A5  1453          .BYTE   BYTE_COUNT_0A           ; byte count for command packet
03  03A6  1454          .BYTE   WRITE                   ; command
00  03A7  1455          .BYTE   MODIFIER_00
00  03A8  1456          .BYTE   UNIT_0
08  03A9  1457          .BYTE   SWITCH_08
00  03AA  1458          .BYTE   SEQUENCE_L
00  03AB  1459          .BYTE   SEQUENCE_H
00  03AC  1460          .BYTE   BYTE_COUNT_00           ; low (data)
02  03AD  1461          .BYTE   BYTE_COUNT_02           ; high (data)
8F  03AE  1462          .BYTE   BLOCK_8F                ; low
00  03AF  1463          .BYTE   BLOCK_00                ; high
94  03B0  1464          .BYTE   CHECKSUM_94             ; low
14  03B1  1465          .BYTE   CHECKSUM_14             ; high
    03B2  1466
    03B2  1467
    03B2  1468 MB_WRITE_7_CMD:
02  03B2  1469          .BYTE   COMMAND_FLAG
0A  03B3  1470          .BYTE   BYTE_COUNT_0A           ; byte count for command packet
03  03B4  1471          .BYTE   WRITE                   ; command
00  03B5  1472          .BYTE   MODIFIER_00
00  03B6  1473          .BYTE   UNIT_0
08  03B7  1474          .BYTE   SWITCH_08
00  03B8  1475          .BYTE   SEQUENCE_L
00  03B9  1476          .BYTE   SEQUENCE_H
01  03BA  1477          .BYTE   BYTE_COUNT_01           ; low (data)
00  03BB  1478          .BYTE   BYTE_COUNT_00           ; high (data)
0F  03BC  1479          .BYTE   BLOCK_0F                ; low
00  03BD  1480          .BYTE   BLOCK_00                ; high
15  03BE  1481          .BYTE   CHECKSUM_15             ; low
12  03BF  1482          .BYTE   CHECKSUM_12             ; high
    03C0  1483
    03C0  1484
    03C0  1485 MB_READ_5_CMD:
02  03C0  1486          .BYTE   COMMAND_FLAG
0A  03C1  1487          .BYTE   BYTE_COUNT_0A           ; byte count for command packet
02  03C2  1488          .BYTE   READ                    ; command
00  03C3  1489          .BYTE   MODIFIER_00
00  03C4  1490          .BYTE   UNIT_0
08  03C5  1491          .BYTE   SWITCH_08
00  03C6  1492          .BYTE   SEQUENCE_L
00  03C7  1493          .BYTE   SEQUENCE_H
00  03C8  1494          .BYTE   BYTE_COUNT_00           ; low (data)
02  03C9  1495          .BYTE   BYTE_COUNT_02           ; high (data)
0F  03CA  1496          .BYTE   BLOCK_0F                ; low
00  03CB  1497          .BYTE   BLOCK_00                ; high
13  03CC  1498          .BYTE   CHECKSUM_13             ; low
14  03CD  1499          .BYTE   CHECKSUM_14             ; high
    03CE  1500
```

```
       03CE   1502 MB_READ_6_CMD:
   02  03CE   1503         .BYTE   COMMAND_FLAG
   0A  03CF   1504         .BYTE   BYTE_COUNT_0A          ; byte count for command packet
   02  03D0   1505         .BYTE   READ                   ; command
   00  03D1   1506         .BYTE   MODIFIER_00
   00  03D2   1507         .BYTE   UNIT_0
   08  03D3   1508         .BYTE   SWITCH_08
   00  03D4   1509         .BYTE   SEQUENCE_L
   00  03D5   1510         .BYTE   SEQUENCE_H
   00  03D6   1511         .BYTE   BYTE_COUNT_00          ; low (data)
   02  03D7   1512         .BYTE   BYTE_COUNT_02          ; high (data)
   8F  03D8   1513         .BYTE   BLOCK_8F               ; low
   00  03D9   1514         .BYTE   BLOCK_00               ; high
   93  03DA   1515         .BYTE   CHECKSUM_93            ; low
   14  03DB   1516         .BYTE   CHECKSUM_14            ; high
       03DC   1517
       03DC   1518
       03DC   1519 MB_WRITE_8_CMD:
   02  03DC   1520         .BYTE   COMMAND_FLAG
   0A  03DD   1521         .BYTE   BYTE_COUNT_0A          ; byte count for command packet
   03  03DE   1522         .BYTE   WRITE                  ; command
   80  03DF   1523         .BYTE   MODIFIER_80
   00  03E0   1524         .BYTE   UNIT_0
   08  03E1   1525         .BYTE   SWITCH_08
   00  03E2   1526         .BYTE   SEQUENCE_L
   00  03E3   1527         .BYTE   SEQUENCE_H
   80  03E4   1528         .BYTE   BYTE_COUNT_80          ; low (data)
   00  03E5   1529         .BYTE   BYTE_COUNT_00          ; high (data)
   0D  03E6   1530         .BYTE   BLOCK_0D               ; low
   00  03E7   1531         .BYTE   BLOCK_00               ; high
   92  03E8   1532         .BYTE   CHECKSUM_92            ; low
   92  03E9   1533         .BYTE   CHECKSUM_92            ; high
       03EA   1534
       03EA   1535
       03EA   1536 MB_WRITE_9_CMD:
   02  03EA   1537         .BYTE   COMMAND_FLAG
   0A  03EB   1538         .BYTE   BYTE_COUNT_0A          ; byte count for command packet
   03  03EC   1539         .BYTE   WRITE                  ; command
   80  03ED   1540         .BYTE   MODIFIER_80
   00  03EE   1541         .BYTE   UNIT_0
   08  03EF   1542         .BYTE   SWITCH_08
   00  03F0   1543         .BYTE   SEQUENCE_L
   00  03F1   1544         .BYTE   SEQUENCE_H
   80  03F2   1545         .BYTE   BYTE_COUNT_80          ; low (data)
   00  03F3   1546         .BYTE   BYTE_COUNT_00          ; high (data)
   0D  03F4   1547         .BYTE   BLOCK_0D               ; low
   02  03F5   1548         .BYTE   BLOCK_02               ; high
   92  03F6   1549         .BYTE   CHECKSUM_92            ; low
   94  03F7   1550         .BYTE   CHECKSUM_94            ; high
       03F8   1551
```

```
     03F8  1553 MB_WRITE_10_CMD:
 02  03F8  1554           .BYTE    COMMAND_FLAG
 0A  03F9  1555           .BYTE    BYTE_COUNT_0A             ; byte count for command packet
 03  03FA  1556           .BYTE    WRITE                     ; command
 80  03FB  1557           .BYTE    MODIFIER_80
 00  03FC  1558           .BYTE    UNIT_0
 08  03FD  1559           .BYTE    SWITCH_08
 00  03FE  1560           .BYTE    SEQUENCE_L
 00  03FF  1561           .BYTE    SEQUENCE_H
 01  0400  1562           .BYTE    BYTE_COUNT_01             ; low (data)
 00  0401  1563           .BYTE    BYTE_COUNT_00             ; high (data)
 0D  0402  1564           .BYTE    BLOCK_0D                  ; low
 00  0403  1565           .BYTE    BLOCK_00                  ; high
 13  0404  1566           .BYTE    CHECKSUM_13               ; low
 92  0405  1567           .BYTE    CHECKSUM_92               ; high
     0406  1568
     0406  1569
     0406  1570 MB_READ_7_CMD:
 02  0406  1571           .BYTE    COMMAND_FLAG
 0A  0407  1572           .BYTE    BYTE_COUNT_0A             ; byte count for command packet
 02  0408  1573           .BYTE    READ                      ; command
 80  0409  1574           .BYTE    MODIFIER_80
 00  040A  1575           .BYTE    UNIT_0
 08  040B  1576           .BYTE    SWITCH_08
 00  040C  1577           .BYTE    SEQUENCE_L
 00  040D  1578           .BYTE    SEQUENCE_H
 80  040E  1579           .BYTE    BYTE_COUNT_80             ; low (data)
 00  040F  1580           .BYTE    BYTE_COUNT_00             ; high (data)
 0D  0410  1581           .BYTE    BLOCK_0D                  ; low
 00  0411  1582           .BYTE    BLOCK_00                  ; high
 91  0412  1583           .BYTE    CHECKSUM_91               ; low
 92  0413  1584           .BYTE    CHECKSUM_92               ; high
     0414  1585
     0414  1586
     0414  1587 MB_READ_8_CMD:
 02  0414  1588           .BYTE    COMMAND_FLAG
 0A  0415  1589           .BYTE    BYTE_COUNT_0A             ; byte count for command packet
 02  0416  1590           .BYTE    READ                      ; command
 80  0417  1591           .BYTE    MODIFIER_80
 00  0418  1592           .BYTE    UNIT_0
 08  0419  1593           .BYTE    SWITCH_08
 00  041A  1594           .BYTE    SEQUENCE_L
 00  041B  1595           .BYTE    SEQUENCE_H
 80  041C  1596           .BYTE    BYTE_COUNT_80             ; low (data)
 00  041D  1597           .BYTE    BYTE_COUNT_00             ; high (data)
 0D  041E  1598           .BYTE    BLOCK_0D                  ; low
 02  041F  1599           .BYTE    BLOCK_02                  ; high
 91  0420  1600           .BYTE    CHECKSUM_91               ; low
 94  0421  1601           .BYTE    CHECKSUM_94               ; high
     0422  1602
     0422  1604 MB_WRITE_11_CMD:
 02  0422  1605          .BYTE   COMMAND_FLAG
 0A  0423  1606          .BYTE   BYTE_COUNT_0A          ; byte count for command packet
 03  0424  1607          .BYTE   WRITE                  ; command
 00  0425  1608          .BYTE   MODIFIER_00
 00  0426  1609          .BYTE   UNIT_0
 08  0427  1610          .BYTE   SWITCH_08
 00  0428  1611          .BYTE   SEQUENCE_L
 00  0429  1612          .BYTE   SEQUENCE_H
 00  042A  1613          .BYTE   BYTE_COUNT_00          ; low (data)
 04  042B  1614          .BYTE   BYTE_COUNT_04          ; high (data)
 00  042C  1615          .BYTE   BLOCK_00               ; low
 00  042D  1616          .BYTE   BLOCK_00               ; high
 05  042E  1617          .BYTE   CHECKSUM_05            ; low
 16  042F  1618          .BYTE   CHECKSUM_16            ; high
     0430  1619
     0430  1620
     0430  1621 MB_READ_9_CMD:
 02  0430  1622          .BYTE   COMMAND_FLAG
 0A  0431  1623          .BYTE   BYTE_COUNT_0A          ; byte count for command packet
 02  0432  1624          .BYTE   READ                   ; command
 00  0433  1625          .BYTE   MODIFIER_00
 00  0434  1626          .BYTE   UNIT_0
 08  0435  1627          .BYTE   SWITCH_08
 00  0436  1628          .BYTE   SEQUENCE_L
 00  0437  1629          .BYTE   SEQUENCE_H
 00  0438  1630          .BYTE   BYTE_COUNT_00          ; low (data)
 04  0439  1631          .BYTE   BYTE_COUNT_04          ; high (data)
 00  043A  1632          .BYTE   BLOCK_00               ; low
 00  043B  1633          .BYTE   BLOCK_00               ; high
 04  043C  1634          .BYTE   CHECKSUM_04            ; low
 16  043D  1635          .BYTE   CHECKSUM_16            ; high
     043E  1636
     043E  1637
     043E  1638 MB_WRITE_12_CMD:
 02  043E  1639          .BYTE   COMMAND_FLAG
 0A  043F  1640          .BYTE   BYTE_COUNT_0A          ; byte count for command packet
 03  0440  1641          .BYTE   WRITE                  ; command
 00  0441  1642          .BYTE   MODIFIER_00
 00  0442  1643          .BYTE   UNIT_0
 08  0443  1644          .BYTE   SWITCH_08
 00  0444  1645          .BYTE   SEQUENCE_L
 00  0445  1646          .BYTE   SEQUENCE_H
 00  0446  1647          .BYTE   BYTE_COUNT_00          ; low (data)
 02  0447  1648          .BYTE   BYTE_COUNT_02          ; high (data)
 80  0448  1649          .BYTE   BLOCK_80               ; low
 01  0449  1650          .BYTE   BLOCK_01               ; high
 85  044A  1651          .BYTE   CHECKSUM_85            ; low
 15  044B  1652          .BYTE   CHECKSUM_15            ; high
     044C  1653
```

```
    044C  1655 MB_READ_10_CMD:
02  044C  1656          .BYTE    COMMAND_FLAG
0A  044D  1657          .BYTE    BYTE_COUNT_0A          ; byte count for command packet
02  044E  1658          .BYTE    READ                   ; command
00  044F  1659          .BYTE    MODIFIER_00
00  0450  1660          .BYTE    UNIT_0
08  0451  1661          .BYTE    SWITCH_08
00  0452  1662          .BYTE    SEQUENCE_L
00  0453  1663          .BYTE    SEQUENCE_H
00  0454  1664          .BYTE    BYTE_COUNT_00          ; low (data)
02  0455  1665          .BYTE    BYTE_COUNT_02          ; high (data)
80  0456  1666          .BYTE    BLOCK_80               ; low
01  0457  1667          .BYTE    BLOCK_01               ; high
84  0458  1668          .BYTE    CHECKSUM_84            ; low
15  0459  1669          .BYTE    CHECKSUM_15            ; high
    045A  1670
    045A  1671
    045A  1672 MB_WRITE_13_CMD:
02  045A  1673          .BYTE    COMMAND_FLAG
0A  045B  1674          .BYTE    BYTE_COUNT_0A          ; byte count for command packet
03  045C  1675          .BYTE    WRITE                  ; command
00  045D  1676          .BYTE    MODIFIER_00
00  045E  1677          .BYTE    UNIT_0
08  045F  1678          .BYTE    SWITCH_08
00  0460  1679          .BYTE    SEQUENCE_L
00  0461  1680          .BYTE    SEQUENCE_H
00  0462  1681          .BYTE    BYTE_COUNT_00          ; low (data)
02  0463  1682          .BYTE    BYTE_COUNT_02          ; high (data)
00  0464  1683          .BYTE    BLOCK_00               ; low
01  0465  1684          .BYTE    BLOCK_01               ; high
05  0466  1685          .BYTE    CHECKSUM_05            ; low
15  0467  1686          .BYTE    CHECKSUM_15            ; high
    0468  1687
    0468  1688
    0468  1689 MB_READ_11_CMD:
02  0468  1690          .BYTE    COMMAND_FLAG
0A  0469  1691          .BYTE    BYTE_COUNT_0A          ; byte count for command packet
02  046A  1692          .BYTE    READ                   ; command
00  046B  1693          .BYTE    MODIFIER_00
00  046C  1694          .BYTE    UNIT_0
08  046D  1695          .BYTE    SWITCH_08
00  046E  1696          .BYTE    SEQUENCE_L
00  046F  1697          .BYTE    SEQUENCE_H
00  0470  1698          .BYTE    BYTE_COUNT_00          ; low (data)
02  0471  1699          .BYTE    BYTE_COUNT_02          ; high (data)
00  0472  1700          .BYTE    BLOCK_00               ; low
01  0473  1701          .BYTE    BLOCK_01               ; high
04  0474  1702          .BYTE    CHECKSUM_04            ; low
15  0475  1703          .BYTE    CHECKSUM_15            ; high
    0476  1704
```

```
        0476  1706 MB_WRITE_14_CMD:
   02   0476  1707          .BYTE    COMMAND_FLAG
   0A   0477  1708          .BYTE    BYTE_COUNT_0A             ; byte count for command packet
   03   0478  1709          .BYTE    WRITE                     ; command
   00   0479  1710          .BYTE    MODIFIER_00
   00   047A  1711          .BYTE    UNIT_0
   08   047B  1712          .BYTE    SWITCH_08
   00   047C  1713          .BYTE    SEQUENCE_L
   00   047D  1714          .BYTE    SEQUENCE_H
   F4   047E  1715          .BYTE    BYTE_COUNT_F4             ; low (data)
   01   047F  1716          .BYTE    BYTE_COUNT_01             ; high (data)
   01   0480  1717          .BYTE    BLOCK_01                  ; low
   01   0481  1718          .BYTE    BLOCK_01                  ; high
   FA   0482  1719          .BYTE    CHECKSUM_FA               ; low
   14   0483  1720          .BYTE    CHECKSUM_14               ; high
        0484  1721
        0484  1722
        0484  1723 MB_READ_12_CMD:
   02   0484  1724          .BYTE    COMMAND_FLAG
   0A   0485  1725          .BYTE    BYTE_COUNT_0A             ; byte count for command packet
   02   0486  1726          .BYTE    READ                      ; command
   00   0487  1727          .BYTE    MODIFIER_00
   00   0488  1728          .BYTE    UNIT_0
   08   0489  1729          .BYTE    SWITCH_08
   00   048A  1730          .BYTE    SEQUENCE_L
   00   048B  1731          .BYTE    SEQUENCE_H
   00   048C  1732          .BYTE    BYTE_COUNT_00             ; low (data)
   02   048D  1733          .BYTE    BYTE_COUNT_02             ; high (data)
   01   048E  1734          .BYTE    BLOCK_01                  ; low
   01   048F  1735          .BYTE    BLOCK_01                  ; high
   05   0490  1736          .BYTE    CHECKSUM_05               ; low
   15   0491  1737          .BYTE    CHECKSUM_15               ; high
        0492  1738
        0492  1739
        0492  1740 MB_WRITE_15_CMD:
   02   0492  1741          .BYTE    COMMAND_FLAG
   0A   0493  1742          .BYTE    BYTE_COUNT_0A             ; byte count for command packet
   03   0494  1743          .BYTE    WRITE                     ; command
   00   0495  1744          .BYTE    MODIFIER_00
   00   0496  1745          .BYTE    UNIT_0
   08   0497  1746          .BYTE    SWITCH_08
   00   0498  1747          .BYTE    SEQUENCE_L
   00   0499  1748          .BYTE    SEQUENCE_H
   00   049A  1749          .BYTE    BYTE_COUNT_00             ; low (data)
   02   049B  1750          .BYTE    BYTE_COUNT_02             ; high (data)
   15   049C  1751          .BYTE    BLOCK_15                  ; low
   00   049D  1752          .BYTE    BLOCK_00                  ; high
   1A   049E  1753          .BYTE    CHECKSUM_1A               ; low
   14   049F  1754          .BYTE    CHECKSUM_14               ; high
        04A0  1755
```

```
     04A0  1757 MB_READ_13_CMD:
02   04A0  1758         .BYTE   COMMAND_FLAG
0A   04A1  1759         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
02   04A2  1760         .BYTE   READ                    ; command
00   04A3  1761         .BYTE   MODIFIER_00
00   04A4  1762         .BYTE   UNIT_0
08   04A5  1763         .BYTE   SWITCH_08
00   04A6  1764         .BYTE   SEQUENCE_L
00   04A7  1765         .BYTE   SEQUENCE_H
00   04A8  1766         .BYTE   BYTE_COUNT_00           ; low (data)
02   04A9  1767         .BYTE   BYTE_COUNT_02           ; high (data)
15   04AA  1768         .BYTE   BLOCK_15                ; low
00   04AB  1769         .BYTE   BLOCK_00                ; high
19   04AC  1770         .BYTE   CHECKSUM_19             ; low
14   04AD  1771         .BYTE   CHECKSUM_14             ; high
     04AE  1772
     04AE  1773
     04AE  1774 MB_WRITE_16_CMD:
02   04AE  1775         .BYTE   COMMAND_FLAG
0A   04AF  1776         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
03   04B0  1777         .BYTE   WRITE                   ; command
00   04B1  1778         .BYTE   MODIFIER_00
00   04B2  1779         .BYTE   UNIT_0
08   04B3  1780         .BYTE   SWITCH_08
00   04B4  1781         .BYTE   SEQUENCE_L
00   04B5  1782         .BYTE   SEQUENCE_H
00   04B6  1783         .BYTE   BYTE_COUNT_00           ; low (data)
02   04B7  1784         .BYTE   BYTE_COUNT_02           ; high (data)
15   04B8  1785         .BYTE   BLOCK_15                ; low
01   04B9  1786         .BYTE   BLOCK_01                ; high
1A   04BA  1787         .BYTE   CHECKSUM_1A             ; low
15   04BB  1788         .BYTE   CHECKSUM_15             ; high
     04BC  1789
     04BC  1790
     04BC  1791 MB_READ_14_CMD:
02   04BC  1792         .BYTE   COMMAND_FLAG
0A   04BD  1793         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
02   04BE  1794         .BYTE   READ                    ; command
00   04BF  1795         .BYTE   MODIFIER_00
00   04C0  1796         .BYTE   UNIT_0
08   04C1  1797         .BYTE   SWITCH_08
00   04C2  1798         .BYTE   SEQUENCE_L
00   04C3  1799         .BYTE   SEQUENCE_H
00   04C4  1800         .BYTE   BYTE_COUNT_00           ; low (data)
02   04C5  1801         .BYTE   BYTE_COUNT_02           ; high (data)
15   04C6  1802         .BYTE   BLOCK_15                ; low
01   04C7  1803         .BYTE   BLOCK_01                ; high
19   04C8  1804         .BYTE   CHECKSUM_19             ; low
15   04C9  1805         .BYTE   CHECKSUM_15             ; high
```

```
        04CA  1807 MB_WRITE_17_CMD:
   02   04CA  1808          .BYTE   COMMAND_FLAG
   0A   04CB  1809          .BYTE   BYTE_COUNT_0A          ; byte count for command packet
   03   04CC  1810          .BYTE   WRITE                  ; command
   00   04CD  1811          .BYTE   MODIFIER_00
   00   04CE  1812          .BYTE   UNIT_0
   08   04CF  1813          .BYTE   SWITCH_08
   00   04D0  1814          .BYTE   SEQUENCE_L
   00   04D1  1815          .BYTE   SEQUENCE_H
   F4   04D2  1816          .BYTE   BYTE_COUNT_F4          ; low (data)
   01   04D3  1817          .BYTE   BYTE_COUNT_01          ; high (data)
   80   04D4  1818          .BYTE   BLOCK_80               ; low
   00   04D5  1819          .BYTE   BLOCK_00               ; high
   79   04D6  1820          .BYTE   CHECKSUM_79            ; low
   14   04D7  1821          .BYTE   CHECKSUM_14            ; high
        04D8  1822
        04D8  1823
        04D8  1824 MB_READ_15_CMD:
   02   04D8  1825          .BYTE   COMMAND_FLAG
   0A   04D9  1826          .BYTE   BYTE_COUNT_0A          ; byte count for command packet
   02   04DA  1827          .BYTE   READ                   ; command
   00   04DB  1828          .BYTE   MODIFIER_00
   00   04DC  1829          .BYTE   UNIT_0
   08   04DD  1830          .BYTE   SWITCH_08
   00   04DE  1831          .BYTE   SEQUENCE_L
   00   04DF  1832          .BYTE   SEQUENCE_H
   00   04E0  1833          .BYTE   BYTE_COUNT_00          ; low (data)
   02   04E1  1834          .BYTE   BYTE_COUNT_02          ; high (data)
   80   04E2  1835          .BYTE   BLOCK_80               ; low
   00   04E3  1836          .BYTE   BLOCK_00               ; high
   84   04E4  1837          .BYTE   CHECKSUM_84            ; low
   14   04E5  1838          .BYTE   CHECKSUM_14            ; high
        04E6  1839
        04E6  1840
        04E6  1841 MB_WRITE_18_CMD:
   02   04E6  1842          .BYTE   COMMAND_FLAG
   0A   04E7  1843          .BYTE   BYTE_COUNT_0A          ; byte count for command packet
   03   04E8  1844          .BYTE   WRITE                  ; command
   80   04E9  1845          .BYTE   MODIFIER_80
   00   04EA  1846          .BYTE   UNIT_0
   08   04EB  1847          .BYTE   SWITCH_08
   00   04EC  1848          .BYTE   SEQUENCE_L
   00   04ED  1849          .BYTE   SEQUENCE_H
   00   04EE  1850          .BYTE   BYTE_COUNT_00          ; low (data)
   01   04EF  1851          .BYTE   BYTE_COUNT_01          ; high (data)
   04   04F0  1852          .BYTE   BLOCK_04               ; low
   02   04F1  1853          .BYTE   BLOCK_02               ; high
   09   04F2  1854          .BYTE   CHECKSUM_09            ; low
   95   04F3  1855          .BYTE   CHECKSUM_95            ; high
```

```
    04F4  1857 MB_READ_16_CMD:
02  04F4  1858          .BYTE    COMMAND_FLAG
0A  04F5  1859          .BYTE    BYTE_COUNT_0A              ; byte count for command packet
02  04F6  1860          .BYTE    READ                       ; command
80  04F7  1861          .BYTE    MODIFIER_80
00  04F8  1862          .BYTE    UNIT_0
08  04F9  1863          .BYTE    SWITCH_08
00  04FA  1864          .BYTE    SEQUENCE_L
00  04FB  1865          .BYTE    SEQUENCE_H
00  04FC  1866          .BYTE    BYTE_COUNT_00              ; low (data)
01  04FD  1867          .BYTE    BYTE_COUNT_01              ; high (data)
04  04FE  1868          .BYTE    BLOCK_04                   ; low
02  04FF  1869          .BYTE    BLOCK_02                   ; high
08  0500  1870          .BYTE    CHECKSUM_08                ; low
95  0501  1871          .BYTE    CHECKSUM_95                ; high
    0502  1872
    0502  1873
    0502  1874 MB_WRITE_19_CMD:
02  0502  1875          .BYTE    COMMAND_FLAG
0A  0503  1876          .BYTE    BYTE_COUNT_0A              ; byte count for command packet
03  0504  1877          .BYTE    WRITE                      ; command
80  0505  1878          .BYTE    MODIFIER_80
00  0506  1879          .BYTE    UNIT_0
08  0507  1880          .BYTE    SWITCH_08
00  0508  1881          .BYTE    SEQUENCE_L
00  0509  1882          .BYTE    SEQUENCE_H
80  050A  1883          .BYTE    BYTE_COUNT_80              ; low (data)
00  050B  1884          .BYTE    BYTE_COUNT_00              ; high (data)
04  050C  1885          .BYTE    BLOCK_04                   ; low
06  050D  1886          .BYTE    BLOCK_06                   ; high
89  050E  1887          .BYTE    CHECKSUM_89                ; low
98  050F  1888          .BYTE    CHECKSUM_98                ; high
    0510  1889
    0510  1890
    0510  1891 MB_READ_17_CMD:
02  0510  1892          .BYTE    COMMAND_FLAG
0A  0511  1893          .BYTE    BYTE_COUNT_0A              ; byte count for command packet
02  0512  1894          .BYTE    READ                       ; command
80  0513  1895          .BYTE    MODIFIER_80
00  0514  1896          .BYTE    UNIT_0
08  0515  1897          .BYTE    SWITCH_08
00  0516  1898          .BYTE    SEQUENCE_L
00  0517  1899          .BYTE    SEQUENCE_H
80  0518  1900          .BYTE    BYTE_COUNT_80              ; low (data)
00  0519  1901          .BYTE    BYTE_COUNT_00              ; high (data)
04  051A  1902          .BYTE    BLOCK_04                   ; low
06  051B  1903          .BYTE    BLOCK_06                   ; high
88  051C  1904          .BYTE    CHECKSUM_88                ; low
98  051D  1905          .BYTE    CHECKSUM_98                ; high
```

```
          051E  1907 MB_WRITE_20_CMD:
    02    051E  1908          .BYTE   COMMAND_FLAG
    0A    051F  1909          .BYTE   BYTE_COUNT_0A              ; byte count for command packet
    03    0520  1910          .BYTE   WRITE                      ; command
    80    0521  1911          .BYTE   MODIFIER_80
    00    0522  1912          .BYTE   UNIT_0
    08    0523  1913          .BYTE   SWITCH_08
    00    0524  1914          .BYTE   SEQUENCE_L
    00    0525  1915          .BYTE   SEQUENCE_H
    80    0526  1916          .BYTE   BYTE_COUNT_80              ; low (data)
    00    0527  1917          .BYTE   BYTE_COUNT_00              ; high (data)
    06    0528  1918          .BYTE   BLOCK_06                   ; low
    06    0529  1919          .BYTE   BLOCK_06                   ; high
    8B    052A  1920          .BYTE   CHECKSUM_8B                ; low
    98    052B  1921          .BYTE   CHECKSUM_98                ; high
          052C  1922
          052C  1923
          052C  1924 MB_READ_18_CMD:
    02    052C  1925          .BYTE   COMMAND_FLAG
    0A    052D  1926          .BYTE   BYTE_COUNT_0A              ; byte count for command packet
    02    052E  1927          .BYTE   READ                       ; command
    80    052F  1928          .BYTE   MODIFIER_80
    00    0530  1929          .BYTE   UNIT_0
    08    0531  1930          .BYTE   SWITCH_08
    00    0532  1931          .BYTE   SEQUENCE_L
    00    0533  1932          .BYTE   SEQUENCE_H
    80    0534  1933          .BYTE   BYTE_COUNT_80              ; low (data)
    00    0535  1934          .BYTE   BYTE_COUNT_00              ; high (data)
    06    0536  1935          .BYTE   BLOCK_06                   ; low
    06    0537  1936          .BYTE   BLOCK_06                   ; high
    8A    0538  1937          .BYTE   CHECKSUM_8A                ; low
    98    0539  1938          .BYTE   CHECKSUM_98                ; high
          053A  1939
          053A  1940
          053A  1941 MB_WRITE_21_CMD:
    02    053A  1942          .BYTE   COMMAND_FLAG
    0A    053B  1943          .BYTE   BYTE_COUNT_0A              ; byte count for command packet
    03    053C  1944          .BYTE   WRITE                      ; command
    80    053D  1945          .BYTE   MODIFIER_80
    00    053E  1946          .BYTE   UNIT_0
    08    053F  1947          .BYTE   SWITCH_08
    00    0540  1948          .BYTE   SEQUENCE_L
    00    0541  1949          .BYTE   SEQUENCE_H
    64    0542  1950          .BYTE   BYTE_COUNT_64              ; low (data)
    00    0543  1951          .BYTE   BYTE_COUNT_00              ; high (data)
    05    0544  1952          .BYTE   BLOCK_05                   ; low
    06    0545  1953          .BYTE   BLOCK_06                   ; high
    6E    0546  1954          .BYTE   CHECKSUM_6E                ; low
    98    0547  1955          .BYTE   CHECKSUM_98                ; high
          0548  1956
```

```
        0548   1958 MB_READ_19_CMD:
   02   0548   1959         .BYTE   COMMAND_FLAG
   0A   0549   1960         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
   02   054A   1961         .BYTE   READ                    ; command
   80   054B   1962         .BYTE   MODIFIER_80
   00   054C   1963         .BYTE   UNIT_0
   08   054D   1964         .BYTE   SWITCH_08
   00   054E   1965         .BYTE   SEQUENCE_L
   00   054F   1966         .BYTE   SEQUENCE_H
   80   0550   1967         .BYTE   BYTE_COUNT_80           ; low (data)
   00   0551   1968         .BYTE   BYTE_COUNT_00           ; high (data)
   05   0552   1969         .BYTE   BLOCK_05                ; low
   06   0553   1970         .BYTE   BLOCK_06                ; high
   89   0554   1971         .BYTE   CHECKSUM_89             ; low
   98   0555   1972         .BYTE   CHECKSUM_98             ; high
        0556   1973
        0556   1974
        0556   1975 MB_WRITE_22_CMD:
   02   0556   1976         .BYTE   COMMAND_FLAG
   0A   0557   1977         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
   03   0558   1978         .BYTE   WRITE                   ; command
   80   0559   1979         .BYTE   MODIFIER_80
   00   055A   1980         .BYTE   UNIT_0
   08   055B   1981         .BYTE   SWITCH_08
   00   055C   1982         .BYTE   SEQUENCE_L
   00   055D   1983         .BYTE   SEQUENCE_H
   80   055E   1984         .BYTE   BYTE_COUNT_80           ; low (data)
   00   055F   1985         .BYTE   BYTE_COUNT_00           ; high (data)
   5A   0560   1986         .BYTE   BLOCK_5A                ; low
   02   0561   1987         .BYTE   BLOCK_02                ; high
   DF   0562   1988         .BYTE   CHECKSUM_DF             ; low
   94   0563   1989         .BYTE   CHECKSUM_94             ; high
        0564   1990
        0564   1991
        0564   1992 MB_READ_20_CMD:
   02   0564   1993         .BYTE   COMMAND_FLAG
   0A   0565   1994         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
   02   0566   1995         .BYTE   READ                    ; command
   80   0567   1996         .BYTE   MODIFIER_80
   00   0568   1997         .BYTE   UNIT_0
   08   0569   1998         .BYTE   SWITCH_08
   00   056A   1999         .BYTE   SEQUENCE_L
   00   056B   2000         .BYTE   SEQUENCE_H
   80   056C   2001         .BYTE   BYTE_COUNT_80           ; low (data)
   00   056D   2002         .BYTE   BYTE_COUNT_00           ; high (data)
   5A   056E   2003         .BYTE   BLOCK_5A                ; low
   02   056F   2004         .BYTE   BLOCK_02                ; high
   DE   0570   2005         .BYTE   CHECKSUM_DE             ; low
   94   0571   2006         .BYTE   CHECKSUM_94             ; high
```

```
     0572  2008 MB_WRITE_23_CMD:
 02  0572  2009         .BYTE   COMMAND_FLAG
 0A  0573  2010         .BYTE   BYTE_COUNT_0A             ; byte count for command packet
 03  0574  2011         .BYTE   WRITE                     ; command
 80  0575  2012         .BYTE   MODIFIER_80
 00  0576  2013         .BYTE   UNIT_0
 08  0577  2014         .BYTE   SWITCH_08
 00  0578  2015         .BYTE   SEQUENCE_L
 00  0579  2016         .BYTE   SEQUENCE_H
 80  057A  2017         .BYTE   BYTE_COUNT_80             ; low (data)
 00  057B  2018         .BYTE   BYTE_COUNT_00             ; high (data)
 5A  057C  2019         .BYTE   BLOCK_5A                  ; low
 06  057D  2020         .BYTE   BLOCK_06                  ; high
 DF  057E  2021         .BYTE   CHECKSUM_DF               ; low
 98  057F  2022         .BYTE   CHECKSUM_98               ; high
     0580  2023
     0580  2024
     0580  2025 MB_READ_21_CMD:
 02  0580  2026         .BYTE   COMMAND_FLAG
 0A  0581  2027         .BYTE   BYTE_COUNT_0A             ; byte count for command packet
 02  0582  2028         .BYTE   READ                      ; command
 80  0583  2029         .BYTE   MODIFIER_80
 00  0584  2030         .BYTE   UNIT_0
 08  0585  2031         .BYTE   SWITCH_08
 00  0586  2032         .BYTE   SEQUENCE_L
 00  0587  2033         .BYTE   SEQUENCE_H
 80  0588  2034         .BYTE   BYTE_COUNT_80             ; low (data)
 00  0589  2035         .BYTE   BYTE_COUNT_00             ; high (data)
 5A  058A  2036         .BYTE   BLOCK_5A                  ; low
 06  058B  2037         .BYTE   BLOCK_06                  ; high
 DE  058C  2038         .BYTE   CHECKSUM_DE               ; low
 98  058D  2039         .BYTE   CHECKSUM_98               ; high
     058E  2040
     058E  2041
     058E  2042 MB_WRITE_24_CMD:
 02  058E  2043         .BYTE   COMMAND_FLAG
 0A  058F  2044         .BYTE   BYTE_COUNT_0A             ; byte count for command packet
 03  0590  2045         .BYTE   WRITE                     ; command
 80  0591  2046         .BYTE   MODIFIER_80
 00  0592  2047         .BYTE   UNIT_0
 08  0593  2048         .BYTE   SWITCH_08
 00  0594  2049         .BYTE   SEQUENCE_L
 00  0595  2050         .BYTE   SEQUENCE_H
 64  0596  2051         .BYTE   BYTE_COUNT_64             ; low (data)
 00  0597  2052         .BYTE   BYTE_COUNT_00             ; high (data)
 06  0598  2053         .BYTE   BLOCK_06                  ; low
 02  0599  2054         .BYTE   BLOCK_02                  ; high
 6F  059A  2055         .BYTE   CHECKSUM_6F               ; low
 94  059B  2056         .BYTE   CHECKSUM_94               ; high
```

```
     059C  2058 MB_READ_22_CMD:
 02  059C  2059         .BYTE   COMMAND_FLAG
 0A  059D  2060         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
 02  059E  2061         .BYTE   READ                    ; command
 80  059F  2062         .BYTE   MODIFIER_80
 00  05A0  2063         .BYTE   UNIT_0
 08  05A1  2064         .BYTE   SWITCH_08
 00  05A2  2065         .BYTE   SEQUENCE_L
 00  05A3  2066         .BYTE   SEQUENCE_H
 80  05A4  2067         .BYTE   BYTE_COUNT_80           ; low (data)
 00  05A5  2068         .BYTE   BYTE_COUNT_00           ; high (data)
 06  05A6  2069         .BYTE   BLOCK_06                ; low
 02  05A7  2070         .BYTE   BLOCK_02                ; high
 8A  05A8  2071         .BYTE   CHECKSUM_8A             ; low
 94  05A9  2072         .BYTE   CHECKSUM_94             ; high
     05AA  2073
     05AA  2074
     05AA  2075 MB_READ_23_CMD:
 02  05AA  2076         .BYTE   COMMAND_FLAG
 0A  05AB  2077         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
 02  05AC  2078         .BYTE   READ                    ; command
 00  05AD  2079         .BYTE   MODIFIER_00
 00  05AE  2080         .BYTE   UNIT_0
 08  05AF  2081         .BYTE   SWITCH_08
 00  05B0  2082         .BYTE   SEQUENCE_L
 00  05B1  2083         .BYTE   SEQUENCE_H
 00  05B2  2084         .BYTE   BYTE_COUNT_00           ; low (data)
 02  05B3  2085         .BYTE   BYTE_COUNT_02           ; high (data)
 00  05B4  2086         .BYTE   BLOCK_00                ; low
 00  05B5  2087         .BYTE   BLOCK_00                ; high
 04  05B6  2088         .BYTE   CHECKSUM_04             ; low
 14  05B7  2089         .BYTE   CHECKSUM_14             ; high
     05B8  2090
     05B8  2091
     05B8  2092
     05B8  2093
     05B8  2094 MB_READ_24_CMD:
 02  05B8  2095         .BYTE   COMMAND_FLAG
 0A  05B9  2096         .BYTE   BYTE_COUNT_0A           ; byte count for command packet
 02  05BA  2097         .BYTE   READ                    ; command
 00  05BB  2098         .BYTE   MODIFIER_00
 00  05BC  2099         .BYTE   UNIT_0
 08  05BD  2100         .BYTE   SWITCH_08
 00  05BE  2101         .BYTE   SEQUENCE_L
 00  05BF  2102         .BYTE   SEQUENCE_H
 00  05C0  2103         .BYTE   BYTE_COUNT_00           ; low (data)
 02  05C1  2104         .BYTE   BYTE_COUNT_02           ; high (data)
 15  05C2  2105         .BYTE   BLOCK_15                ; low
 00  05C3  2106         .BYTE   BLOCK_00                ; high
 19  05C4  2107         .BYTE   CHECKSUM_19             ; low
 14  05C5  2108         .BYTE   CHECKSUM_14             ; high
     05C6  2109
```

```
    05C6  2111 MB_POS_1_CMD:
02  05C6  2112          .BYTE    COMMAND_FLAG
0A  05C7  2113          .BYTE    BYTE_COUNT_0A           ; byte count for command packet
05  05C8  2114          .BYTE    POSITION                ; command
00  05C9  2115          .BYTE    MODIFIER_00
00  05CA  2116          .BYTE    UNIT_0
08  05CB  2117          .BYTE    SWITCH_08
00  05CC  2118          .BYTE    SEQUENCE_L
00  05CD  2119          .BYTE    SEQUENCE_H
00  05CE  2120          .BYTE    BYTE_COUNT_00           ; low (data)
00  05CF  2121          .BYTE    BYTE_COUNT_00           ; high (data)
15  05D0  2122          .BYTE    BLOCK_15                ; low
00  05D1  2123          .BYTE    BLOCK_00                ; high
1C  05D2  2124          .BYTE    CHECKSUM_1C             ; low
12  05D3  2125          .BYTE    CHECKSUM_12             ; high
    05D4  2126
    05D4  2127
    05D4  2128
    05D4  2129
```

```
     05D4  2131 .SBTTL  END PACKET TABLE
     05D4  2132
     05D4  2133 ; end packets for RSP
     05D4  2134
     05D4  2135 B_END_PACKET_1:
 02  05D4  2136          .BYTE   END_FLAG
 0A  05D5  2137          .BYTE   BYTE_COUNT_0A          ; byte count for end packet
 40  05D6  2138          .BYTE   END_CODE               ; op code
 00  05D7  2139          .BYTE   SUCCESS_CODE
 00  05D8  2140          .BYTE   UNIT_0
 00  05D9  2141          .BYTE   NOT_USED
 00  05DA  2142          .BYTE   SEQUENCE_L
 00  05DB  2143          .BYTE   SEQUENCE_H
 00  05DC  2144          .BYTE   BYTE_COUNT_00          ; low (data)
 00  05DD  2145          .BYTE   BYTE_COUNT_00          ; high (data)
 00  05DE  2146          .BYTE   SUMMARY_STAT_L
 00  05DF  2147          .BYTE   SUMMARY_STAT_H
 42  05E0  2148          .BYTE   CHECKSUM_42            ; low
 0A  05E1  2149          .BYTE   CHECKSUM_0A            ; high
     05E2  2150
     05E2  2151
     05E2  2152 B_END_PACKET_2:
 02  05E2  2153          .BYTE   END_FLAG
 0A  05E3  2154          .BYTE   BYTE_COUNT_0A          ; byte count for end packet
 40  05E4  2155          .BYTE   END_CODE               ; op code
 00  05E5  2156          .BYTE   SUCCESS_CODE
 00  05E6  2157          .BYTE   UNIT_0
 00  05E7  2158          .BYTE   NOT_USED
 00  05E8  2159          .BYTE   SEQUENCE_L
 00  05E9  2160          .BYTE   SEQUENCE_H
 80  05EA  2161          .BYTE   BYTE_COUNT_80          ; low (data)
 00  05EB  2162          .BYTE   BYTE_COUNT_00          ; high (data)
 00  05EC  2163          .BYTE   SUMMARY_STAT_L
 00  05ED  2164          .BYTE   SUMMARY_STAT_H
 C2  05EE  2165          .BYTE   CHECKSUM_C2            ; low
 0A  05EF  2166          .BYTE   CHECKSUM_0A            ; high
     05F0  2167
     05F0  2168
     05F0  2169 B_END_PACKET_3:
 02  05F0  2170          .BYTE   END_FLAG
 0A  05F1  2171          .BYTE   BYTE_COUNT_0A          ; byte count for end packet
 40  05F2  2172          .BYTE   END_CODE               ; op code
 00  05F3  2173          .BYTE   SUCCESS_CODE
 00  05F4  2174          .BYTE   UNIT_0
 00  05F5  2175          .BYTE   NOT_USED
 00  05F6  2176          .BYTE   SEQUENCE_L
 00  05F7  2177          .BYTE   SEQUENCE_H
 1A  05F8  2178          .BYTE   BYTE_COUNT_1A          ; low (data)
 00  05F9  2179          .BYTE   BYTE_COUNT_00          ; high (data)
 00  05FA  2180          .BYTE   SUMMARY_STAT_L
 00  05FB  2181          .BYTE   SUMMARY_STAT_H
 5C  05FC  2182          .BYTE   CHECKSUM_5C            ; low
 0A  05FD  2183          .BYTE   CHECKSUM_0A            ; high
     05FE  2184
```

```
        05FE  2186 B_END_PACKET_4:
   02   05FE  2187          .BYTE   END_FLAG
   0A   05FF  2188          .BYTE   BYTE_COUNT_0A           ; byte count for end packet
   40   0600  2189          .BYTE   END_CODE                ; op code
   00   0601  2190          .BYTE   SUCCESS_CODE
   00   0602  2191          .BYTE   UNIT_0
   00   0603  2192          .BYTE   NOT_USED
   00   0604  2193          .BYTE   SEQUENCE_L
   00   0605  2194          .BYTE   SEQUENCE_H
   00   0606  2195          .BYTE   BYTE_COUNT_00           ; low (data)
   02   0607  2196          .BYTE   BYTE_COUNT_02           ; high (data)
   00   0608  2197          .BYTE   SUMMARY_STAT_L
   00   0609  2198          .BYTE   SUMMARY_STAT_H
   42   060A  2199          .BYTE   CHECKSUM_42             ; low
   0C   060B  2200          .BYTE   CHECKSUM_0C             ; high
        060C  2201
        060C  2202
        060C  2203 B_END_PACKET_5:
   02   060C  2204          .BYTE   END_FLAG
   0A   060D  2205          .BYTE   BYTE_COUNT_0A           ; byte count for end packet
   40   060E  2206          .BYTE   END_CODE                ; op code
   00   060F  2207          .BYTE   SUCCESS_CODE
   00   0610  2208          .BYTE   UNIT_0
   00   0611  2209          .BYTE   NOT_USED
   00   0612  2210          .BYTE   SEQUENCE_L
   00   0613  2211          .BYTE   SEQUENCE_H
   80   0614  2212          .BYTE   BYTE_COUNT_80           ; low (data)
   00   0615  2213          .BYTE   BYTE_COUNT_00           ; high (data)
   00   0616  2214          .BYTE   SUMMARY_STAT_L
   00   0617  2215          .BYTE   SUMMARY_STAT_H
   C2   0618  2216          .BYTE   CHECKSUM_C2             ; low
   0A   0619  2217          .BYTE   CHECKSUM_0A             ; high
        061A  2218
        061A  2219
        061A  2220 B_END_PACKET_6:
   02   061A  2221          .BYTE   END_FLAG
   0A   061B  2222          .BYTE   BYTE_COUNT_0A           ; byte count for end packet
   40   061C  2223          .BYTE   END_CODE                ; op code
   00   061D  2224          .BYTE   SUCCESS_CODE
   00   061E  2225          .BYTE   UNIT_0
   00   061F  2226          .BYTE   NOT_USED
   00   0620  2227          .BYTE   SEQUENCE_L
   00   0621  2228          .BYTE   SEQUENCE_H
   01   0622  2229          .BYTE   BYTE_COUNT_01           ; low (data)
   00   0623  2230          .BYTE   BYTE_COUNT_00           ; high (data)
   00   0624  2231          .BYTE   SUMMARY_STAT_L
   00   0625  2232          .BYTE   SUMMARY_STAT_H
   43   0626  2233          .BYTE   CHECKSUM_43             ; low
   0A   0627  2234          .BYTE   CHECKSUM_0A             ; high
        0628  2235
        0628   2237 B_END_PACKET_7:
    02  0628   2238         .BYTE    END_FLAG
    0A  0629   2239         .BYTE    BYTE_COUNT_0A              ; byte count for end packet
    40  062A   2240         .BYTE    END_CODE                   ; op code
    00  062B   2241         .BYTE    SUCCESS_CODE
    00  062C   2242         .BYTE    UNIT_0
    00  062D   2243         .BYTE    NOT_USED
    00  062E   2244         .BYTE    SEQUENCE_L
    00  062F   2245         .BYTE    SEQUENCE_H
    00  0630   2246         .BYTE    BYTE_COUNT_00              ; low (data)
    04  0631   2247         .BYTE    BYTE_COUNT_04              ; high (data)
    00  0632   2248         .BYTE    SUMMARY_STAT_L
    00  0633   2249         .BYTE    SUMMARY_STAT_H
    42  0634   2250         .BYTE    CHECKSUM_42                ; low
    0E  0635   2251         .BYTE    CHECKSUM_0E                ; high
        0636   2252
        0636   2253
        0636   2254 B_END_PACKET_8:
    02  0636   2255         .BYTE    END_FLAG
    0A  0637   2256         .BYTE    BYTE_COUNT_0A              ; byte count for end packet
    40  0638   2257         .BYTE    END_CODE                   ; op code
    00  0639   2258         .BYTE    SUCCESS_CODE
    00  063A   2259         .BYTE    UNIT_0
    00  063B   2260         .BYTE    NOT_USED
    00  063C   2261         .BYTE    SEQUENCE_L
    00  063D   2262         .BYTE    SEQUENCE_H
    F4  063E   2263         .BYTE    BYTE_COUNT_F4              ; low (data)
    01  063F   2264         .BYTE    BYTE_COUNT_01              ; high (data)
    00  0640   2265         .BYTE    SUMMARY_STAT_L
    00  0641   2266         .BYTE    SUMMARY_STAT_H
    36  0642   2267         .BYTE    CHECKSUM_36                ; low
    0C  0643   2268         .BYTE    CHECKSUM_0C                ; high
        0644   2269
        0644   2270 B_END_PACKET_9:
    02  0644   2271         .BYTE    END_FLAG
    0A  0645   2272         .BYTE    BYTE_COUNT_0A              ; byte count for end packet
    40  0646   2273         .BYTE    END_CODE                   ; op code
    00  0647   2274         .BYTE    SUCCESS_CODE
    00  0648   2275         .BYTE    UNIT_0
    00  0649   2276         .BYTE    NOT_USED
    00  064A   2277         .BYTE    SEQUENCE_L
    00  064B   2278         .BYTE    SEQUENCE_H
    00  064C   2279         .BYTE    BYTE_COUNT_00              ; low (data)
    01  064D   2280         .BYTE    BYTE_COUNT_01              ; high (data)
    00  064E   2281         .BYTE    SUMMARY_STAT_L
    00  064F   2282         .BYTE    SUMMARY_STAT_H
    42  0650   2283         .BYTE    CHECKSUM_42                ; low
    0B  0651   2284         .BYTE    CHECKSUM_0B                ; high
        0652   2285
```

```
         0652   2287 B_END_PACKET_10:
   02    0652   2288            .BYTE    END_FLAG
   0A    0653   2289            .BYTE    BYTE_COUNT_0A               ; byte count for end packet
   40    0654   2290            .BYTE    END_CODE                    ; op code
   00    0655   2291            .BYTE    SUCCESS_CODE
   00    0656   2292            .BYTE    UNIT_0
   00    0657   2293            .BYTE    NOT_USED
   00    0658   2294            .BYTE    SEQUENCE_L
   00    0659   2295            .BYTE    SEQUENCE_H
   64    065A   2296            .BYTE    BYTE_COUNT_64               ; low (data)
   00    065B   2297            .BYTE    BYTE_COUNT_00               ; high (data)
   00    065C   2298            .BYTE    SUMMARY_STAT_L
   00    065D   2299            .BYTE    SUMMARY_STAT_H
   A6    065E   2300            .BYTE    CHECKSUM_A6                 ; low
   0A    065F   2301            .BYTE    CHECKSUM_0A                 ; high
         0660   2302
         0660   2303 ; end packets for MRSP
         0660   2304
         0660   2305 MB_END_PACKET_1:
   02    0660   2306            .BYTE    END_FLAG
   0A    0661   2307            .BYTE    BYTE_COUNT_0A               ; byte count for end packet
   40    0662   2308            .BYTE    END_CODE                    ; op code
   00    0663   2309            .BYTE    SUCCESS_CODE
   00    0664   2310            .BYTE    UNIT_0
   08    0665   2311            .BYTE    SWITCH_08
   00    0666   2312            .BYTE    SEQUENCE_L
   00    0667   2313            .BYTE    SEQUENCE_H
   00    0668   2314            .BYTE    BYTE_COUNT_00               ; low (data)
   00    0669   2315            .BYTE    BYTE_COUNT_00               ; high (data)
   00    066A   2316            .BYTE    SUMMARY_STAT_L
   00    066B   2317            .BYTE    SUMMARY_STAT_H
   42    066C   2318            .BYTE    CHECKSUM_42                 ; low
   12    066D   2319            .BYTE    CHECKSUM_12                 ; high
         066E   2320
         066E   2321
         066E   2322 MB_END_PACKET_2:
   02    066E   2323            .BYTE    END_FLAG
   0A    066F   2324            .BYTE    BYTE_COUNT_0A               ; byte count for end packet
   40    0670   2325            .BYTE    END_CODE                    ; op code
   00    0671   2326            .BYTE    SUCCESS_CODE
   00    0672   2327            .BYTE    UNIT_0
   08    0673   2328            .BYTE    SWITCH_08
   00    0674   2329            .BYTE    SEQUENCE_L
   00    0675   2330            .BYTE    SEQUENCE_H
   80    0676   2331            .BYTE    BYTE_COUNT_80               ; low (data)
   00    0677   2332            .BYTE    BYTE_COUNT_00               ; high (data)
   00    0678   2333            .BYTE    SUMMARY_STAT_L
   00    0679   2334            .BYTE    SUMMARY_STAT_H
   C2    067A   2335            .BYTE    CHECKSUM_C2                 ; low
   12    067B   2336            .BYTE    CHECKSUM_12                 ; high
         067C   2337
         067C   2338
         067C   2339 MB_END_PACKET_3:
   02    067C   2340            .BYTE    END_FLAG
   0A    067D   2341            .BYTE    BYTE_COUNT_0A               ; byte count for end packet
```

```
 40  067E  2342          .BYTE   END_CODE                ; op code
 00  067F  2343          .BYTE   SUCCESS_CODE
 00  0680  2344          .BYTE   UNIT_0
 08  0681  2345          .BYTE   SWITCH_08
 00  0682  2346          .BYTE   SEQUENCE_L
 00  0683  2347          .BYTE   SEQUENCE_H
 1A  0684  2348          .BYTE   BYTE_COUNT_1A           ; low (data)
 00  0685  2349          .BYTE   BYTE_COUNT_00           ; high (data)
 00  0686  2350          .BYTE   SUMMARY_STAT_L
 00  0687  2351          .BYTE   SUMMARY_STAT_H
 5C  0688  2352          .BYTE   CHECKSUM_5C             ; low
 12  0689  2353          .BYTE   CHECKSUM_12             ; high
     068A  2354
```

```
      068A  2356 MB_END_PACKET_4:
  02  068A  2357          .BYTE    END_FLAG
  0A  068B  2358          .BYTE    BYTE_COUNT_0A           ; byte count for end packet
  40  068C  2359          .BYTE    END_CODE                ; op code
  00  068D  2360          .BYTE    SUCCESS_CODE
  00  068E  2361          .BYTE    UNIT_0
  08  068F  2362          .BYTE    SWITCH_08
  00  0690  2363          .BYTE    SEQUENCE_L
  00  0691  2364          .BYTE    SEQUENCE_H
  00  0692  2365          .BYTE    BYTE_COUNT_00           ; low (data)
  02  0693  2366          .BYTE    BYTE_COUNT_02           ; high (data)
  00  0694  2367          .BYTE    SUMMARY_STAT_L
  00  0695  2368          .BYTE    SUMMARY_STAT_H
  42  0696  2369          .BYTE    CHECKSUM_42             ; low
  14  0697  2370          .BYTE    CHECKSUM_14             ; high
      0698  2371
      0698  2372
      0698  2373 MB_END_PACKET_5:
  02  0698  2374          .BYTE    END_FLAG
  0A  0699  2375          .BYTE    BYTE_COUNT_0A           ; byte count for end packet
  40  069A  2376          .BYTE    END_CODE                ; op code
  00  069B  2377          .BYTE    SUCCESS_CODE
  00  069C  2378          .BYTE    UNIT_0
  08  069D  2379          .BYTE    SWITCH_08
  00  069E  2380          .BYTE    SEQUENCE_L
  00  069F  2381          .BYTE    SEQUENCE_H
  80  06A0  2382          .BYTE    BYTE_COUNT_80           ; low (data)
  00  06A1  2383          .BYTE    BYTE_COUNT_00           ; high (data)
  00  06A2  2384          .BYTE    SUMMARY_STAT_L
  00  06A3  2385          .BYTE    SUMMARY_STAT_H
  C2  06A4  2386          .BYTE    CHECKSUM_C2             ; low
  12  06A5  2387          .BYTE    CHECKSUM_12             ; high
      06A6  2388
      06A6  2389
      06A6  2390 MB_END_PACKET_6:
  02  06A6  2391          .BYTE    END_FLAG
  0A  06A7  2392          .BYTE    BYTE_COUNT_0A           ; byte count for end packet
  40  06A8  2393          .BYTE    END_CODE                ; op code
  00  06A9  2394          .BYTE    SUCCESS_CODE
  00  06AA  2395          .BYTE    UNIT_0
  08  06AB  2396          .BYTE    SWITCH_08
  00  06AC  2397          .BYTE    SEQUENCE_L
  00  06AD  2398          .BYTE    SEQUENCE_H
  01  06AE  2399          .BYTE    BYTE_COUNT_01           ; low (data)
  00  06AF  2400          .BYTE    BYTE_COUNT_00           ; high (data)
  00  06B0  2401          .BYTE    SUMMARY_STAT_L
  00  06B1  2402          .BYTE    SUMMARY_STAT_H
  43  06B2  2403          .BYTE    CHECKSUM_43             ; low
  12  06B3  2404          .BYTE    CHECKSUM_12             ; high
      06B4  2405
```

```
    06B4  2407 MB_END_PACKET_7:
02  06B4  2408          .BYTE   END_FLAG
0A  06B5  2409          .BYTE   BYTE_COUNT_0A           ; byte count for end packet
40  06B6  2410          .BYTE   END_CODE                ; op code
00  06B7  2411          .BYTE   SUCCESS_CODE
00  06B8  2412          .BYTE   UNIT_0
08  06B9  2413          .BYTE   SWITCH_08
00  06BA  2414          .BYTE   SEQUENCE_L
00  06BB  2415          .BYTE   SEQUENCE_H
00  06BC  2416          .BYTE   BYTE_COUNT_00           ; low (data)
04  06BD  2417          .BYTE   BYTE_COUNT_04           ; high (data)
00  06BE  2418          .BYTE   SUMMARY_STAT_L
00  06BF  2419          .BYTE   SUMMARY_STAT_H
42  06C0  2420          .BYTE   CHECKSUM_42             ; low
16  06C1  2421          .BYTE   CHECKSUM_16             ; high
    06C2  2422
    06C2  2423
    06C2  2424 MB_END_PACKET_8:
02  06C2  2425          .BYTE   END_FLAG
0A  06C3  2426          .BYTE   BYTE_COUNT_0A           ; byte count for end packet
40  06C4  2427          .BYTE   END_CODE                ; op code
00  06C5  2428          .BYTE   SUCCESS_CODE
00  06C6  2429          .BYTE   UNIT_0
08  06C7  2430          .BYTE   SWITCH_08
00  06C8  2431          .BYTE   SEQUENCE_L
00  06C9  2432          .BYTE   SEQUENCE_H
F4  06CA  2433          .BYTE   BYTE_COUNT_F4           ; low (data)
01  06CB  2434          .BYTE   BYTE_COUNT_01           ; high (data)
00  06CC  2435          .BYTE   SUMMARY_STAT_L
00  06CD  2436          .BYTE   SUMMARY_STAT_H
36  06CE  2437          .BYTE   CHECKSUM_36             ; low
14  06CF  2438          .BYTE   CHECKSUM_14             ; high
    06D0  2439
    06D0  2440 MB_END_PACKET_9:
02  06D0  2441          .BYTE   END_FLAG
0A  06D1  2442          .BYTE   BYTE_COUNT_0A           ; byte count for end packet
40  06D2  2443          .BYTE   END_CODE                ; op code
00  06D3  2444          .BYTE   SUCCESS_CODE
00  06D4  2445          .BYTE   UNIT_0
08  06D5  2446          .BYTE   SWITCH_08
00  06D6  2447          .BYTE   SEQUENCE_L
00  06D7  2448          .BYTE   SEQUENCE_H
00  06D8  2449          .BYTE   BYTE_COUNT_00           ; low (data)
01  06D9  2450          .BYTE   BYTE_COUNT_01           ; high (data)
00  06DA  2451          .BYTE   SUMMARY_STAT_L
00  06DB  2452          .BYTE   SUMMARY_STAT_H
42  06DC  2453          .BYTE   CHECKSUM_42             ; low
13  06DD  2454          .BYTE   CHECKSUM_13             ; high
    06DE  2455
```

```
     06DE  2457 MB_END_PACKET_10:
 02  06DE  2458         .BYTE   END_FLAG
 0A  06DF  2459         .BYTE   BYTE_COUNT_0A          ; byte count for end packet
 40  06E0  2460         .BYTE   END_CODE               ; op code
 00  06E1  2461         .BYTE   SUCCESS_CODE
 00  06E2  2462         .BYTE   UNIT_0
 08  06E3  2463         .BYTE   SWITCH_08
 00  06E4  2464         .BYTE   SEQUENCE_L
 00  06E5  2465         .BYTE   SEQUENCE_H
 64  06E6  2466         .BYTE   BYTE_COUNT_64          ; low (data)
 00  06E7  2467         .BYTE   BYTE_COUNT_00          ; high (data)
 00  06E8  2468         .BYTE   SUMMARY_STAT_L
 00  06E9  2469         .BYTE   SUMMARY_STAT_H
 A6  06EA  2470         .BYTE   CHECKSUM_A6            ; low
 12  06EB  2471         .BYTE   CHECKSUM_12            ; high
     06EC  2472
```

```
                                  06EC  2474 .SBTTL  DATA PACKET TABLE
                                  06EC  2475
                                  06EC  2476 B_DATA_PKT_1:
                              01  06EC  2477         .BYTE   DATA_FLAG
                              1A  06ED  2478         .BYTE   BYTE_COUNT_1A   ; byte count in hex.
                            0004  06EE  2479         .WORD   DATA_0004
                            0000  06F0  2480         .WORD   DATA_0000       ; this data is used to write
                            0001  06F2  2481         .WORD   DATA_0001       ;  one directory area on the
                            0000  06F4  2482         .WORD   DATA_0000       ;  tape called `SCRAT.CH'
                            000F  06F6  2483         .WORD   DATA_000F
                            0400  06F8  2484         .WORD   DATA_0400
                            774A  06FA  2485         .WORD   DATA_774A
                            0960  06FC  2486         .WORD   DATA_0960
                            1400  06FE  2487         .WORD   DATA_1400
                            0000  0700  2488         .WORD   DATA_0000
                            0000  0702  2489         .WORD   DATA_0000
                            0000  0704  2490         .WORD   DATA_0000
                            0800  0706  2491         .WORD   DATA_0800
                              BF  0708  2492         .BYTE   CHECKSUM_BF     ; low
                              BA  0709  2493         .BYTE   CHECKSUM_BA     ; high
                                  070A  2494
                                  070A  2495
                                  070A  2496 B_DATA_PKT_2:
                              01  070A  2497         .BYTE   DATA_FLAG
                              80  070B  2498         .BYTE   BYTE_COUNT_80   ; byte count in hex.
 00FF'00FF'00FF'00FF'00FF'00FF'00FF'  070C  2499         .WORD   DATA_00FF[^X40] ; data and number of words in hex.
 00FF'00FF'00FF'00FF'00FF'00FF'00FF'  071A
 00FF'00FF'00FF'00FF'00FF'00FF'00FF'  0728
 00FF'00FF'00FF'00FF'00FF'00FF'00FF'  0736
 00FF'00FF'00FF'00FF'00FF'00FF'00FF'  0744
 00FF'00FF'00FF'00FF'00FF'00FF'00FF'  0752
 00FF'00FF'00FF'00FF'00FF'00FF'00FF'  0760
 00FF'00FF'00FF'00FF'00FF'00FF'00FF'  076E
 00FF'00FF'00FF'00FF'00FF'00FF'00FF'  077C
                           00FF'  078A
                              C1  078C  2500         .BYTE   CHECKSUM_C1     ; low
                              BF  078D  2501         .BYTE   CHECKSUM_BF     ; high
                                  078E  2502
                                  078E  2503
                                  078E  2504 B_DATA_PKT_3:
                              01  078E  2505         .BYTE   DATA_FLAG
                              80  078F  2506         .BYTE   BYTE_COUNT_80   ; byte count in hex.
 FFFF'FFFF'FFFF'FFFF'FFFF'FFFF'FFFF'  0790  2507         .WORD   DATA_FFFF[^X40] ; data and number of words in hex.
 FFFF'FFFF'FFFF'FFFF'FFFF'FFFF'FFFF'  079E
 FFFF'FFFF'FFFF'FFFF'FFFF'FFFF'FFFF'  07AC
 FFFF'FFFF'FFFF'FFFF'FFFF'FFFF'FFFF'  07BA
 FFFF'FFFF'FFFF'FFFF'FFFF'FFFF'FFFF'  07C8
 FFFF'FFFF'FFFF'FFFF'FFFF'FFFF'FFFF'  07D6
 FFFF'FFFF'FFFF'FFFF'FFFF'FFFF'FFFF'  07E4
 FFFF'FFFF'FFFF'FFFF'FFFF'FFFF'FFFF'  07F2
 FFFF'FFFF'FFFF'FFFF'FFFF'FFFF'FFFF'  0800
                           FFFF'  080E
                              01  0810  2508         .BYTE   CHECKSUM_01     ; low
                              80  0811  2509         .BYTE   CHECKSUM_80     ; high
                                  0812  2510
```

```
                                    0812  2511
                                    0812  2512 B_DATA_PKT_4:
                                01  0812  2513          .BYTE   DATA_FLAG
                                01  0813  2514          .BYTE   BYTE_COUNT_01    ; byte count in hex.
                                AA  0814  2515          .BYTE   DATA_00AA        ; data and number of words in hex.
                                AB  0815  2516          .BYTE   CHECKSUM_AB      ; low
                                01  0816  2517          .BYTE   CHECKSUM_01      ; high
                                    0817  2518
                                    0817  2519
                                    0817  2520 B_DATA_PKT_5:
                                01  0817  2521          .BYTE   DATA_FLAG
                                80  0818  2522          .BYTE   BYTE_COUNT_80    ; byte count in hex.
                              00AA  0819  2523          .WORD   DATA_00AA
 0000'0000'0000'0000'0000'0000'0000'  081B  2524          .WORD   DATA_0000[^X3F] ; data and number of words in hex.
 0000'0000'0000'0000'0000'0000'0000'  0829
 0000'0000'0000'0000'0000'0000'0000'  0837
 0000'0000'0000'0000'0000'0000'0000'  0845
 0000'0000'0000'0000'0000'0000'0000'  0853
 0000'0000'0000'0000'0000'0000'0000'  0861
 0000'0000'0000'0000'0000'0000'0000'  086F
 0000'0000'0000'0000'0000'0000'0000'  087D
 0000'0000'0000'0000'0000'0000'0000'  088B
                                AB  0899  2525          .BYTE   CHECKSUM_AB      ; low
                                80  089A  2526          .BYTE   CHECKSUM_80      ; high
```

```
                                     089B  2528 B_DATA_PKT_6:
                                  01 089B  2529         .BYTE   DATA_FLAG
                                  80 089C  2530         .BYTE   BYTE_COUNT_80   ; byte count in hex.
 0000'0000'0000'0000'0000'0000'0000' 089D  2531         .WORD   DATA_0000[^X40] ; data and number of words in hex.
 0000'0000'0000'0000'0000'0000'0000' 08AB
 0000'0000'0000'0000'0000'0000'0000' 08B9
 0000'0000'0000'0000'0000'0000'0000' 08C7
 0000'0000'0000'0000'0000'0000'0000' 08D5
 0000'0000'0000'0000'0000'0000'0000' 08E3
 0000'0000'0000'0000'0000'0000'0000' 08F1
 0000'0000'0000'0000'0000'0000'0000' 08FF
 0000'0000'0000'0000'0000'0000'0000' 090D
                               0000' 091B
                                  01 091D  2532         .BYTE   CHECKSUM_01     ; low
                                  80 091E  2533         .BYTE   CHECKSUM_80     ; high
                                     091F  2534
                                     091F  2535
                                     091F  2536 B_DATA_PKT_7:
                                  01 091F  2537         .BYTE   DATA_FLAG
                                  80 0920  2538         .BYTE   BYTE_COUNT_80   ; byte count in hex.
 0001'0001'0001'0001'0001'0001'0001' 0921  2539         .WORD   DATA_0001[^X40] ; data and number of words in hex.
 0001'0001'0001'0001'0001'0001'0001' 092F
 0001'0001'0001'0001'0001'0001'0001' 093D
 0001'0001'0001'0001'0001'0001'0001' 094B
 0001'0001'0001'0001'0001'0001'0001' 0959
 0001'0001'0001'0001'0001'0001'0001' 0967
 0001'0001'0001'0001'0001'0001'0001' 0975
 0001'0001'0001'0001'0001'0001'0001' 0983
 0001'0001'0001'0001'0001'0001'0001' 0991
                               0001' 099F
                                  41 09A1  2540         .BYTE   CHECKSUM_41     ; low
                                  80 09A2  2541         .BYTE   CHECKSUM_80     ; high
                                     09A3  2542
                                     09A3  2543
                                     09A3  2544 B_DATA_PKT_8:
                                  01 09A3  2545         .BYTE   DATA_FLAG
                                  80 09A4  2546         .BYTE   BYTE_COUNT_80   ; byte count in hex.
 0180'0180'0180'0180'0180'0180'0180' 09A5  2547         .WORD   DATA_0180[^X40] ; data and number of words in hex.
 0180'0180'0180'0180'0180'0180'0180' 09B3
 0180'0180'0180'0180'0180'0180'0180' 09C1
 0180'0180'0180'0180'0180'0180'0180' 09CF
 0180'0180'0180'0180'0180'0180'0180' 09DD
 0180'0180'0180'0180'0180'0180'0180' 09EB
 0180'0180'0180'0180'0180'0180'0180' 09F9
 0180'0180'0180'0180'0180'0180'0180' 0A07
 0180'0180'0180'0180'0180'0180'0180' 0A15
                               0180' 0A23
                                  01 0A25  2548         .BYTE   CHECKSUM_01     ; low
                                  E0 0A26  2549         .BYTE   CHECKSUM_E0     ; high
                                     0A27  2550
                                     0A27  2551
                                     0A27  2552 B_DATA_PKT_9:
                                  01 0A27  2553         .BYTE   DATA_FLAG
                                  80 0A28  2554         .BYTE   BYTE_COUNT_80   ; byte count in hex.
 0100'0100'0100'0100'0100'0100'0100' 0A29  2555         .WORD   DATA_0100[^X40] ; data and number of words in hex.
```

```
0100'0100'0100'0100'0100'0100'0100'  0A37
0100'0100'0100'0100'0100'0100'0100'  0A45
0100'0100'0100'0100'0100'0100'0100'  0A53
0100'0100'0100'0100'0100'0100'0100'  0A61
0100'0100'0100'0100'0100'0100'0100'  0A6F
0100'0100'0100'0100'0100'0100'0100'  0A7D
0100'0100'0100'0100'0100'0100'0100'  0A8B
0100'0100'0100'0100'0100'0100'0100'  0A99
                              0100'  0AA7
                                01   0AA9  2556         .BYTE   CHECKSUM_01     ; low
                                C0   0AAA  2557         .BYTE   CHECKSUM_C0     ; high
                                     0AAB  2558
                                     0AAB  2559
                                     0AAB  2560 B_DATA_PKT_10:
                                01   0AAB  2561         .BYTE   DATA_FLAG
                                80   0AAC  2562         .BYTE   BYTE_COUNT_80   ; byte count in hex.
0101'0101'0101'0101'0101'0101'0101'  0AAD  2563         .WORD   DATA_0101[^X40] ; data and number of words in hex.
0101'0101'0101'0101'0101'0101'0101'  0ABB
0101'0101'0101'0101'0101'0101'0101'  0AC9
0101'0101'0101'0101'0101'0101'0101'  0AD7
0101'0101'0101'0101'0101'0101'0101'  0AE5
0101'0101'0101'0101'0101'0101'0101'  0AF3
0101'0101'0101'0101'0101'0101'0101'  0B01
0101'0101'0101'0101'0101'0101'0101'  0B0F
0101'0101'0101'0101'0101'0101'0101'  0B1D
                              0101'  0B2B
                                41   0B2D  2564         .BYTE   CHECKSUM_41     ; low
                                C0   0B2E  2565         .BYTE   CHECKSUM_C0     ; high
                                     0B2F  2566
                                     0B2F  2567
                                     0B2F  2568 B_DATA_PKT_11:
                                01   0B2F  2569         .BYTE   DATA_FLAG
                                74   0B30  2570         .BYTE   BYTE_COUNT_74   ; byte count in hex.
0101'0101'0101'0101'0101'0101'0101'  0B31  2571         .WORD   DATA_0101[^X3A] ; data and number of words in hex.
0101'0101'0101'0101'0101'0101'0101'  0B3F
0101'0101'0101'0101'0101'0101'0101'  0B4D
0101'0101'0101'0101'0101'0101'0101'  0B5B
0101'0101'0101'0101'0101'0101'0101'  0B69
0101'0101'0101'0101'0101'0101'0101'  0B77
0101'0101'0101'0101'0101'0101'0101'  0B85
0101'0101'0101'0101'0101'0101'0101'  0B93
                         0101'0101'  0BA1
                                75   0BA5  2572         .BYTE   CHECKSUM_75     ; low
                                E8   0BA6  2573         .BYTE   CHECKSUM_E8     ; high
                                     0BA7  2574
```

```
                                      0BA7   2576 B_DATA_PKT_12:
                                  01  0BA7   2577         .BYTE   DATA_FLAG
                                  80  0BA8   2578         .BYTE   BYTE_COUNT_80    ; byte count in hex.
 0101'0101'0101'0101'0101'0101'0101'  0BA9   2579         .WORD   DATA_0101[^X3A] ; data and number of words in hex.
 0101'0101'0101'0101'0101'0101'0101'  0BB7
 0101'0101'0101'0101'0101'0101'0101'  0BC5
 0101'0101'0101'0101'0101'0101'0101'  0BD3
 0101'0101'0101'0101'0101'0101'0101'  0BE1
 0101'0101'0101'0101'0101'0101'0101'  0BEF
 0101'0101'0101'0101'0101'0101'0101'  0BFD
 0101'0101'0101'0101'0101'0101'0101'  0C0B
                           0101'0101'  0C19
      0000'0000'0000'0000'0000'0000'  0C1D   2580         .WORD   DATA_0000[^X06]
                                  75  0C29   2581         .BYTE   CHECKSUM_75      ; low
                                  F4  0C2A   2582         .BYTE   CHECKSUM_F4      ; high
                                      0C2B   2583
                                      0C2B   2584
                                      0C2B   2585 B_DATA_PKT_13:
                                  01  0C2B   2586         .BYTE   DATA_FLAG
                                  80  0C2C   2587         .BYTE   BYTE_COUNT_80    ; byte count in hex.
 0015'0015'0015'0015'0015'0015'0015'  0C2D   2588         .WORD   DATA_0015[^X40] ; data and number of words in hex.
 0015'0015'0015'0015'0015'0015'0015'  0C3B
 0015'0015'0015'0015'0015'0015'0015'  0C49
 0015'0015'0015'0015'0015'0015'0015'  0C57
 0015'0015'0015'0015'0015'0015'0015'  0C65
 0015'0015'0015'0015'0015'0015'0015'  0C73
 0015'0015'0015'0015'0015'0015'0015'  0C81
 0015'0015'0015'0015'0015'0015'0015'  0C8F
 0015'0015'0015'0015'0015'0015'0015'  0C9D
                               0015'  0CAB
                                  41  0CAD   2589         .BYTE   CHECKSUM_41      ; low
                                  85  0CAE   2590         .BYTE   CHECKSUM_85      ; high
                                      0CAF   2591
                                      0CAF   2592
                                      0CAF   2593 B_DATA_PKT_14:
                                  01  0CAF   2594         .BYTE   DATA_FLAG
                                  80  0CB0   2595         .BYTE   BYTE_COUNT_80    ; byte count in hex.
 0115'0115'0115'0115'0115'0115'0115'  0CB1   2596         .WORD   DATA_0115[^X40] ; data and number of words in hex.
 0115'0115'0115'0115'0115'0115'0115'  0CBF
 0115'0115'0115'0115'0115'0115'0115'  0CCD
 0115'0115'0115'0115'0115'0115'0115'  0CDB
 0115'0115'0115'0115'0115'0115'0115'  0CE9
 0115'0115'0115'0115'0115'0115'0115'  0CF7
 0115'0115'0115'0115'0115'0115'0115'  0D05
 0115'0115'0115'0115'0115'0115'0115'  0D13
 0115'0115'0115'0115'0115'0115'0115'  0D21
                               0115'  0D2F
                                  41  0D31   2597         .BYTE   CHECKSUM_41      ; low
                                  C5  0D32   2598         .BYTE   CHECKSUM_C5      ; high
                                      0D33   2599
                                      0D33   2600
                                      0D33   2601 B_DATA_PKT_15:
                                  01  0D33   2602         .BYTE   DATA_FLAG
                                  80  0D34   2603         .BYTE   BYTE_COUNT_80    ; byte count in hex.
 0080'0080'0080'0080'0080'0080'0080'  0D35   2604         .WORD   DATA_0080[^X40] ; data and number of words in hex.
 0080'0080'0080'0080'0080'0080'0080'  0D43
 0080'0080'0080'0080'0080'0080'0080'  0D51
 0080'0080'0080'0080'0080'0080'0080'  0D5F
 0080'0080'0080'0080'0080'0080'0080'  0D6D
 0080'0080'0080'0080'0080'0080'0080'  0D7B
 0080'0080'0080'0080'0080'0080'0080'  0D89
 0080'0080'0080'0080'0080'0080'0080'  0D97
 0080'0080'0080'0080'0080'0080'0080'  0DA5
                              0080'  0DB3
                                01  0DB5  2605          .BYTE    CHECKSUM_01      ; low
                                AE  0DB6  2606          .BYTE    CHECKSUM_AE      ; high
                                    0DB7  2607
                                    0DB7  2608
                                    0DB7  2609 B_DATA_PKT_16:
                                01  0DB7  2610          .BYTE    DATA_FLAG
                                74  0DB8  2611          .BYTE    BYTE_COUNT_74    ; byte count in hex.
 0080'0080'0080'0080'0080'0080'0080'  0DB9  2612          .WORD    DATA_0080[^X3A] ; data and number of words in hex.
 0080'0080'0080'0080'0080'0080'0080'  0DC7
 0080'0080'0080'0080'0080'0080'0080'  0DD5
 0080'0080'0080'0080'0080'0080'0080'  0DE3
 0080'0080'0080'0080'0080'0080'0080'  0DF1
 0080'0080'0080'0080'0080'0080'0080'  0DFF
 0080'0080'0080'0080'0080'0080'0080'  0E0D
 0080'0080'0080'0080'0080'0080'0080'  0E1B
                         0080'0080'  0E29
      0000'0000'0000'0000'0000'0000'  0E2D  2613          .WORD    DATA_0000[^X06]
                                01  0E39  2614          .BYTE    CHECKSUM_01      ; low
                                AE  0E3A  2615          .BYTE    CHECKSUM_AE      ; high
                                    0E3B  2616
                                    0E3B  2617
                                    0E3B  2618 B_DATA_PKT_17:
                                01  0E3B  2619          .BYTE    DATA_FLAG
                                80  0E3C  2620          .BYTE    BYTE_COUNT_80    ; byte count in hex.
 0080'0080'0080'0080'0080'0080'0080'  0E3D  2621          .WORD    DATA_0080[^X3A] ; data and number of words in hex.
 0080'0080'0080'0080'0080'0080'0080'  0E4B
 0080'0080'0080'0080'0080'0080'0080'  0E59
 0080'0080'0080'0080'0080'0080'0080'  0E67
 0080'0080'0080'0080'0080'0080'0080'  0E75
 0080'0080'0080'0080'0080'0080'0080'  0E83
 0080'0080'0080'0080'0080'0080'0080'  0E91
 0080'0080'0080'0080'0080'0080'0080'  0E9F
                         0080'0080'  0EAD
      0000'0000'0000'0000'0000'0000'  0EB1  2622          .WORD    DATA_0000[^X06]
                                01  0EBD  2623          .BYTE    CHECKSUM_01      ; low
                                BA  0EBE  2624          .BYTE    CHECKSUM_BA      ; high
                                    0EBF  2625
```

```
                                            0EBF   2627 B_DATA_PKT_18:
                                       01   0EBF   2628          .BYTE   DATA_FLAG
                                       80   0EC0   2629          .BYTE   BYTE_COUNT_80    ; byte count in hex.
 0204'0204'0204'0204'0204'0204'0204'  0EC1   2630          .WORD   DATA_0204[^X40] ; data and number of words in hex.
 0204'0204'0204'0204'0204'0204'0204'  0ECF
 0204'0204'0204'0204'0204'0204'0204'  0EDD
 0204'0204'0204'0204'0204'0204'0204'  0EEB
 0204'0204'0204'0204'0204'0204'0204'  0EF9
 0204'0204'0204'0204'0204'0204'0204'  0F07
 0204'0204'0204'0204'0204'0204'0204'  0F15
 0204'0204'0204'0204'0204'0204'0204'  0F23
 0204'0204'0204'0204'0204'0204'0204'  0F31
                                     0204'  0F3F
                                       02   0F41   2631          .BYTE   CHECKSUM_02      ; low
                                       01   0F42   2632          .BYTE   CHECKSUM_01      ; high
                                            0F43   2633
                                            0F43   2634
                                            0F43   2635 B_DATA_PKT_19:
                                       01   0F43   2636          .BYTE   DATA_FLAG
                                       80   0F44   2637          .BYTE   BYTE_COUNT_80    ; byte count in hex.
 0604'0604'0604'0604'0604'0604'0604'  0F45   2638          .WORD   DATA_0604[^X40] ; data and number of words in hex.
 0604'0604'0604'0604'0604'0604'0604'  0F53
 0604'0604'0604'0604'0604'0604'0604'  0F61
 0604'0604'0604'0604'0604'0604'0604'  0F6F
 0604'0604'0604'0604'0604'0604'0604'  0F7D
 0604'0604'0604'0604'0604'0604'0604'  0F8B
 0604'0604'0604'0604'0604'0604'0604'  0F99
 0604'0604'0604'0604'0604'0604'0604'  0FA7
 0604'0604'0604'0604'0604'0604'0604'  0FB5
                                     0604'  0FC3
                                       03   0FC5   2639          .BYTE   CHECKSUM_03      ; low
                                       01   0FC6   2640          .BYTE   CHECKSUM_01      ; high
                                            0FC7   2641
                                            0FC7   2642
                                            0FC7   2643 B_DATA_PKT_20:
                                       01   0FC7   2644          .BYTE   DATA_FLAG
                                       80   0FC8   2645          .BYTE   BYTE_COUNT_80    ; byte count in hex.
 0606'0606'0606'0606'0606'0606'0606'  0FC9   2646          .WORD   DATA_0606[^X40] ; data and number of words in hex.
 0606'0606'0606'0606'0606'0606'0606'  0FD7
 0606'0606'0606'0606'0606'0606'0606'  0FE5
 0606'0606'0606'0606'0606'0606'0606'  0FF3
 0606'0606'0606'0606'0606'0606'0606'  1001
 0606'0606'0606'0606'0606'0606'0606'  100F
 0606'0606'0606'0606'0606'0606'0606'  101D
 0606'0606'0606'0606'0606'0606'0606'  102B
 0606'0606'0606'0606'0606'0606'0606'  1039
                                     0606'  1047
                                       83   1049   2647          .BYTE   CHECKSUM_83      ; low
                                       01   104A   2648          .BYTE   CHECKSUM_01      ; high
                                            104B   2649
                                            104B   2650
                                            104B   2651 B_DATA_PKT_21:
                                       01   104B   2652          .BYTE   DATA_FLAG
                                       64   104C   2653          .BYTE   BYTE_COUNT_64    ; byte count in hex.
 0605'0605'0605'0605'0605'0605'0605'  104D   2654          .WORD   DATA_0605[^X32] ; data and number of words in hex.
```

```
0605'0605'0605'0605'0605'0605'0605' 105B
0605'0605'0605'0605'0605'0605'0605' 1069
0605'0605'0605'0605'0605'0605'0605' 1077
0605'0605'0605'0605'0605'0605'0605' 1085
0605'0605'0605'0605'0605'0605'0605' 1093
0605'0605'0605'0605'0605'0605'0605' 10A1
                              0605' 10AF
                                FC  10B1  2655          .BYTE   CHECKSUM_FC     ; low
                                72  10B2  2656          .BYTE   CHECKSUM_72     ; high
                                    10B3  2657
                                    10B3  2658
                                    10B3  2659 B_DATA_PKT_22:
                                01  10B3  2660          .BYTE   DATA_FLAG
                                80  10B4  2661          .BYTE   BYTE_COUNT_80   ; byte count in hex.
0605'0605'0605'0605'0605'0605'0605' 10B5  2662          .WORD   DATA_0605[^X40] ; data and number of words in hex.
0605'0605'0605'0605'0605'0605'0605' 10C3
0605'0605'0605'0605'0605'0605'0605' 10D1
0605'0605'0605'0605'0605'0605'0605' 10DF
0605'0605'0605'0605'0605'0605'0605' 10ED
0605'0605'0605'0605'0605'0605'0605' 10FB
0605'0605'0605'0605'0605'0605'0605' 1109
0605'0605'0605'0605'0605'0605'0605' 1117
0605'0605'0605'0605'0605'0605'0605' 1125
                              0605' 1133
                                F7  1135  2663          .BYTE   CHECKSUM_F7     ; low
                                11  1136  2664          .BYTE   CHECKSUM_11     ; high
                                    1137  2665
                                    1137  2666
                                    1137  2667 B_DATA_PKT_23:
                                01  1137  2668          .BYTE   DATA_FLAG
                                80  1138  2669          .BYTE   BYTE_COUNT_80   ; byte count in hex.
025A'025A'025A'025A'025A'025A'025A' 1139  2670          .WORD   DATA_025A[^X40] ; data and number of words in hex.
025A'025A'025A'025A'025A'025A'025A' 1147
025A'025A'025A'025A'025A'025A'025A' 1155
025A'025A'025A'025A'025A'025A'025A' 1163
025A'025A'025A'025A'025A'025A'025A' 1171
025A'025A'025A'025A'025A'025A'025A' 117F
025A'025A'025A'025A'025A'025A'025A' 118D
025A'025A'025A'025A'025A'025A'025A' 119B
025A'025A'025A'025A'025A'025A'025A' 11A9
                              025A' 11B7
                                82  11B9  2671          .BYTE   CHECKSUM_82     ; low
                                16  11BA  2672          .BYTE   CHECKSUM_16     ; high
                                    11BB  2673
```

```
                                 11BB  2675 B_DATA_PKT_24:
                             01  11BB  2676          .BYTE    DATA_FLAG
                             80  11BC  2677          .BYTE    BYTE_COUNT_80    ; byte count in hex.
 065A'065A'065A'065A'065A'065A'065A'  11BD  2678          .WORD    DATA_065A[^X40] ; data and number of words in hex.
 065A'065A'065A'065A'065A'065A'065A'  11CB
 065A'065A'065A'065A'065A'065A'065A'  11D9
 065A'065A'065A'065A'065A'065A'065A'  11E7
 065A'065A'065A'065A'065A'065A'065A'  11F5
 065A'065A'065A'065A'065A'065A'065A'  1203
 065A'065A'065A'065A'065A'065A'065A'  1211
 065A'065A'065A'065A'065A'065A'065A'  121F
 065A'065A'065A'065A'065A'065A'065A'  122D
                              065A'  123B
                             83  123D  2679          .BYTE    CHECKSUM_83      ; low
                             16  123E  2680          .BYTE    CHECKSUM_16      ; high
                                 123F  2681
                                 123F  2682
                                 123F  2683 B_DATA_PKT_25:
                             01  123F  2684          .BYTE    DATA_FLAG
                             64  1240  2685          .BYTE    BYTE_COUNT_64    ; byte count in hex.
 0206'0206'0206'0206'0206'0206'0206'  1241  2686          .WORD    DATA_0206[^X32] ; data and number of words in hex.
 0206'0206'0206'0206'0206'0206'0206'  124F
 0206'0206'0206'0206'0206'0206'0206'  125D
 0206'0206'0206'0206'0206'0206'0206'  126B
 0206'0206'0206'0206'0206'0206'0206'  1279
 0206'0206'0206'0206'0206'0206'0206'  1287
 0206'0206'0206'0206'0206'0206'0206'  1295
                              0206'  12A3
                             2D  12A5  2687          .BYTE    CHECKSUM_2D      ; low
                             C9  12A6  2688          .BYTE    CHECKSUM_C9      ; high
                                 12A7  2689
                                 12A7  2690
                                 12A7  2691 B_DATA_PKT_26:
                             01  12A7  2692          .BYTE    DATA_FLAG
                             80  12A8  2693          .BYTE    BYTE_COUNT_80    ; byte count in hex.
 0206'0206'0206'0206'0206'0206'0206'  12A9  2694          .WORD    DATA_0206[^X32] ; data and number of words in hex.
 0206'0206'0206'0206'0206'0206'0206'  12B7
 0206'0206'0206'0206'0206'0206'0206'  12C5
 0206'0206'0206'0206'0206'0206'0206'  12D3
 0206'0206'0206'0206'0206'0206'0206'  12E1
 0206'0206'0206'0206'0206'0206'0206'  12EF
 0206'0206'0206'0206'0206'0206'0206'  12FD
                              0206'  130B
 0000'0000'0000'0000'0000'0000'0000'  130D  2695          .WORD    DATA_0000[^X0E]
 0000'0000'0000'0000'0000'0000'0000'  131B
                             2D  1329  2696          .BYTE    CHECKSUM_2D      ; low
                             E5  132A  2697          .BYTE    CHECKSUM_E5      ; high
```

```
                     132B  2699 .SBTTL  COMMAND TABLES
                     132B  2700
                     132B  2701 ; commands for RSP
                     132B  2702
                     132B  2703 ; This is the list of command, end and data packets for the  write block
                     132B  2704 ; padded zeros  subtest. This table lists the addresses that point to
                     132B  2705 ; the actual packet that is needed.
                     132B  2706
           000000BE' 132B  2707 COMMAND_TBL_1:  .ADDRESS        B_WRITE_5_CMD
           0000078E' 132F  2708                 .ADDRESS        B_DATA_PKT_3
           0000078E' 1333  2709                 .ADDRESS        B_DATA_PKT_3
           0000078E' 1337  2710                 .ADDRESS        B_DATA_PKT_3
           0000078E' 133B  2711                 .ADDRESS        B_DATA_PKT_3
           000005FE' 133F  2712                 .ADDRESS        B_END_PACKET_4
                     1343  2713
           000000CC' 1343  2714                 .ADDRESS        B_WRITE_6_CMD
           0000078E' 1347  2715                 .ADDRESS        B_DATA_PKT_3
           0000078E' 134B  2716                 .ADDRESS        B_DATA_PKT_3
           0000078E' 134F  2717                 .ADDRESS        B_DATA_PKT_3
           0000078E' 1353  2718                 .ADDRESS        B_DATA_PKT_3
           000005FE' 1357  2719                 .ADDRESS        B_END_PACKET_4
                     135B  2720
           000000DA' 135B  2721                 .ADDRESS        B_WRITE_7_CMD
           00000812' 135F  2722                 .ADDRESS        B_DATA_PKT_4
           0000061A' 1363  2723                 .ADDRESS        B_END_PACKET_6
                     1367  2724
           000000E8' 1367  2725                 .ADDRESS        B_READ_5_CMD
           00000817' 136B  2726                 .ADDRESS        B_DATA_PKT_5
           0000089B' 136F  2727                 .ADDRESS        B_DATA_PKT_6
           0000089B' 1373  2728                 .ADDRESS        B_DATA_PKT_6
           0000089B' 1377  2729                 .ADDRESS        B_DATA_PKT_6
           000005FE' 137B  2730                 .ADDRESS        B_END_PACKET_4
                     137F  2731
           000000F6' 137F  2732                 .ADDRESS        B_READ_6_CMD
           0000078E' 1383  2733                 .ADDRESS        B_DATA_PKT_3
           0000078E' 1387  2734                 .ADDRESS        B_DATA_PKT_3
           0000078E' 138B  2735                 .ADDRESS        B_DATA_PKT_3
           0000078E' 138F  2736                 .ADDRESS        B_DATA_PKT_3
           000005FE' 1393  2737                 .ADDRESS        B_END_PACKET_4
                     1397  2738
                     1397  2739 ; This is the packet list for the  write record padded zeros  subtest.
                     1397  2740
           00000104' 1397  2741 COMMAND_TBL_2:  .ADDRESS        B_WRITE_8_CMD
           0000078E' 139B  2742                 .ADDRESS        B_DATA_PKT_3
           0000060C' 139F  2743                 .ADDRESS        B_END_PACKET_5
                     13A3  2744
           00000112' 13A3  2745                 .ADDRESS        B_WRITE_9_CMD
           0000078E' 13A7  2746                 .ADDRESS        B_DATA_PKT_3
           0000060C' 13AB  2747                 .ADDRESS        B_END_PACKET_5
                     13AF  2748
           00000120' 13AF  2749                 .ADDRESS        B_WRITE_10_CMD
           00000812' 13B3  2750                 .ADDRESS        B_DATA_PKT_4
           0000061A' 13B7  2751                 .ADDRESS        B_END_PACKET_6
                     13BB  2752
           0000012E' 13BB  2753                 .ADDRESS        B_READ_7_CMD
```

```
00000817' 13BF  2754                    .ADDRESS        B_DATA_PKT_5
0000060C' 13C3  2755                    .ADDRESS        B_END_PACKET_5
          13C7  2756
0000013C' 13C7  2757                    .ADDRESS        B_READ_8_CMD
0000078E' 13CB  2758                    .ADDRESS        B_DATA_PKT_3
0000060C' 13CF  2759                    .ADDRESS        B_END_PACKET_5
          13D3  2760
          13D3  2761 ; This is the packet list for the  block write read  subtest.
          13D3  2762
0000014A' 13D3  2763 COMMAND_TBL_3:  .ADDRESS        B_WRITE_11_CMD
0000089B' 13D7  2764                    .ADDRESS        B_DATA_PKT_6
0000089B' 13DB  2765                    .ADDRESS        B_DATA_PKT_6
0000089B' 13DF  2766                    .ADDRESS        B_DATA_PKT_6
0000089B' 13E3  2767                    .ADDRESS        B_DATA_PKT_6
0000091F' 13E7  2768                    .ADDRESS        B_DATA_PKT_7
0000091F' 13EB  2769                    .ADDRESS        B_DATA_PKT_7
0000091F' 13EF  2770                    .ADDRESS        B_DATA_PKT_7
0000091F' 13F3  2771                    .ADDRESS        B_DATA_PKT_7
00000628' 13F7  2772                    .ADDRESS        B_END_PACKET_7
          13FB  2773
00000158' 13FB  2774                    .ADDRESS        B_READ_9_CMD
0000089B' 13FF  2775                    .ADDRESS        B_DATA_PKT_6
0000089B' 1403  2776                    .ADDRESS        B_DATA_PKT_6
0000089B' 1407  2777                    .ADDRESS        B_DATA_PKT_6
0000089B' 140B  2778                    .ADDRESS        B_DATA_PKT_6
0000091F' 140F  2779                    .ADDRESS        B_DATA_PKT_7
0000091F' 1413  2780                    .ADDRESS        B_DATA_PKT_7
0000091F' 1417  2781                    .ADDRESS        B_DATA_PKT_7
0000091F' 141B  2782                    .ADDRESS        B_DATA_PKT_7
00000628' 141F  2783                    .ADDRESS        B_END_PACKET_7
          1423  2784
00000166' 1423  2785                    .ADDRESS        B_WRITE_12_CMD
000009A3' 1427  2786                    .ADDRESS        B_DATA_PKT_8
000009A3' 142B  2787                    .ADDRESS        B_DATA_PKT_8
000009A3' 142F  2788                    .ADDRESS        B_DATA_PKT_8
000009A3' 1433  2789                    .ADDRESS        B_DATA_PKT_8
000005FE' 1437  2790                    .ADDRESS        B_END_PACKET_4
          143B  2791
00000174' 143B  2792                    .ADDRESS        B_READ_10_CMD
000009A3' 143F  2793                    .ADDRESS        B_DATA_PKT_8
000009A3' 1443  2794                    .ADDRESS        B_DATA_PKT_8
000009A3' 1447  2795                    .ADDRESS        B_DATA_PKT_8
000009A3' 144B  2796                    .ADDRESS        B_DATA_PKT_8
000005FE' 144F  2797                    .ADDRESS        B_END_PACKET_4
          1453  2798
00000182' 1453  2799                    .ADDRESS        B_WRITE_13_CMD
00000A27' 1457  2800                    .ADDRESS        B_DATA_PKT_9
00000A27' 145B  2801                    .ADDRESS        B_DATA_PKT_9
00000A27' 145F  2802                    .ADDRESS        B_DATA_PKT_9
00000A27' 1463  2803                    .ADDRESS        B_DATA_PKT_9
000005FE' 1467  2804                    .ADDRESS        B_END_PACKET_4
          146B  2805
00000190' 146B  2806                    .ADDRESS        B_READ_11_CMD
00000A27' 146F  2807                    .ADDRESS        B_DATA_PKT_9
00000A27' 1473  2808                    .ADDRESS        B_DATA_PKT_9
```

```
                         00000A27' 1477  2809                  .ADDRESS        B_DATA_PKT_9
                         00000A27' 147B  2810                  .ADDRESS        B_DATA_PKT_9
                         000005FE' 147F  2811                  .ADDRESS        B_END_PACKET_4
                                   1483  2812
                         000001BA' 1483  2813                  .ADDRESS        B_WRITE_15_CMD
                         00000C2B' 1487  2814                  .ADDRESS        B_DATA_PKT_13
                         00000C2B' 148B  2815                  .ADDRESS        B_DATA_PKT_13
                         00000C2B' 148F  2816                  .ADDRESS        B_DATA_PKT_13
                         00000C2B' 1493  2817                  .ADDRESS        B_DATA_PKT_13
                         000005FE' 1497  2818                  .ADDRESS        B_END_PACKET_4
                                   149B  2819
                         000001C8' 149B  2820                  .ADDRESS        B_READ_13_CMD
                         00000C2B' 149F  2821                  .ADDRESS        B_DATA_PKT_13
                         00000C2B' 14A3  2822                  .ADDRESS        B_DATA_PKT_13
                         00000C2B' 14A7  2823                  .ADDRESS        B_DATA_PKT_13
                         00000C2B' 14AB  2824                  .ADDRESS        B_DATA_PKT_13
                         000005FE' 14AF  2825                  .ADDRESS        B_END_PACKET_4
                                   14B3  2826
                         000001D6' 14B3  2827                  .ADDRESS        B_WRITE_16_CMD
                         00000CAF' 14B7  2828                  .ADDRESS        B_DATA_PKT_14
                         00000CAF' 14BB  2829                  .ADDRESS        B_DATA_PKT_14
                         00000CAF' 14BF  2830                  .ADDRESS        B_DATA_PKT_14
                         00000CAF' 14C3  2831                  .ADDRESS        B_DATA_PKT_14
                         000005FE' 14C7  2832                  .ADDRESS        B_END_PACKET_4
                                   14CB  2833
                         000001E4' 14CB  2834                  .ADDRESS        B_READ_14_CMD
                         00000CAF' 14CF  2835                  .ADDRESS        B_DATA_PKT_14
                         00000CAF' 14D3  2836                  .ADDRESS        B_DATA_PKT_14
                         00000CAF' 14D7  2837                  .ADDRESS        B_DATA_PKT_14
                         00000CAF' 14DB  2838                  .ADDRESS        B_DATA_PKT_14
                         000005FE' 14DF  2839                  .ADDRESS        B_END_PACKET_4
                                   14E3  2840
                                   14E3  2841 ; This is the packet list for the 'record write read  subtest
                                   14E3  2842
                         0000020E' 14E3  2843 COMMAND_TBL_4:  .ADDRESS        B_WRITE_18_CMD
                         00000EBF' 14E7  2844                  .ADDRESS        B_DATA_PKT_18
                         00000EBF' 14EB  2845                  .ADDRESS        B_DATA_PKT_18
                         00000644' 14EF  2846                  .ADDRESS        B_END_PACKET_9
                                   14F3  2847
                         0000021C' 14F3  2848                  .ADDRESS        B_READ_16_CMD
                         00000EBF' 14F7  2849                  .ADDRESS        B_DATA_PKT_18
                         00000EBF' 14FB  2850                  .ADDRESS        B_DATA_PKT_18
                         00000644' 14FF  2851                  .ADDRESS        B_END_PACKET_9
                                   1503  2852
                         0000022A' 1503  2853                  .ADDRESS        B_WRITE_19_CMD
                         00000F43' 1507  2854                  .ADDRESS        B_DATA_PKT_19
                         0000060C' 150B  2855                  .ADDRESS        B_END_PACKET_5
                                   150F  2856
                         00000238' 150F  2857                  .ADDRESS        B_READ_17_CMD
                         00000F43' 1513  2858                  .ADDRESS        B_DATA_PKT_19
                         0000060C' 1517  2859                  .ADDRESS        B_END_PACKET_5
                                   151B  2860
                         00000246' 151B  2861                  .ADDRESS        B_WRITE_20_CMD
                         00000FC7' 151F  2862                  .ADDRESS        B_DATA_PKT_20
                         0000060C' 1523  2863                  .ADDRESS        B_END_PACKET_5
```

```
                     1527  2864
           00000254' 1527  2865                   .ADDRESS        B_READ_18_CMD
           00000FC7' 152B  2866                   .ADDRESS        B_DATA_PKT_20
           0000060C' 152F  2867                   .ADDRESS        B_END_PACKET_5
                     1533  2868
           0000027E' 1533  2869                   .ADDRESS        B_WRITE_22_CMD
           00001137' 1537  2870                   .ADDRESS        B_DATA_PKT_23
           0000060C' 153B  2871                   .ADDRESS        B_END_PACKET_5
                     153F  2872
           0000028C' 153F  2873                   .ADDRESS        B_READ_20_CMD
           00001137' 1543  2874                   .ADDRESS        B_DATA_PKT_23
           0000060C' 1547  2875                   .ADDRESS        B_END_PACKET_5
                     154B  2876
           0000029A' 154B  2877                   .ADDRESS        B_WRITE_23_CMD
           000011BB' 154F  2878                   .ADDRESS        B_DATA_PKT_24
           0000060C' 1553  2879                   .ADDRESS        B_END_PACKET_5
                     1557  2880
           000002A8' 1557  2881                   .ADDRESS        B_READ_21_CMD
           000011BB' 155B  2882                   .ADDRESS        B_DATA_PKT_24
           0000060C' 155F  2883                   .ADDRESS        B_END_PACKET_5
                     1563  2884
                     1563  2885
           000002D2' 1563  2886 COMMAND_TBL_5:    .ADDRESS        B_READ_23_CMD
           0000089B' 1567  2887                   .ADDRESS        B_DATA_PKT_6
           0000089B' 156B  2888                   .ADDRESS        B_DATA_PKT_6
           0000089B' 156F  2889                   .ADDRESS        B_DATA_PKT_6
           0000089B' 1573  2890                   .ADDRESS        B_DATA_PKT_6
           000005FE' 1577  2891                   .ADDRESS        B_END_PACKET_4
                     157B  2892
           000002E0' 157B  2893                   .ADDRESS        B_READ_24_CMD
           00000C2B' 157F  2894                   .ADDRESS        B_DATA_PKT_13
           00000C2B' 1583  2895                   .ADDRESS        B_DATA_PKT_13
           00000C2B' 1587  2896                   .ADDRESS        B_DATA_PKT_13
           00000C2B' 158B  2897                   .ADDRESS        B_DATA_PKT_13
           000005FE' 158F  2898                   .ADDRESS        B_END_PACKET_4
                     1593  2899
           000002EE' 1593  2900                   .ADDRESS        B_POS_1_CMD
           000005D4' 1597  2901                   .ADDRESS        B_END_PACKET_1
                     159B  2902
           000002D2' 159B  2903                   .ADDRESS        B_READ_23_CMD
           0000089B' 159F  2904                   .ADDRESS        B_DATA_PKT_6
           0000089B' 15A3  2905                   .ADDRESS        B_DATA_PKT_6
           0000089B' 15A7  2906                   .ADDRESS        B_DATA_PKT_6
           0000089B' 15AB  2907                   .ADDRESS        B_DATA_PKT_6
           000005FE' 15AF  2908                   .ADDRESS        B_END_PACKET_4
                     15B3  2909
                     15B3  2910
                     15B3  2911 ; commands for MRSP
                     15B3  2912
                     15B3  2913
                     15B3  2914 ; This is the list of command, end and data packets for the  write block
                     15B3  2915 ; padded zeros' subtest. This table lists the addresses that point to
                     15B3  2916 ; the actual packet that is needed.
                     15B3  2917
           00000396' 15B3  2918 MCOMMAND_TBL_1: .ADDRESS          MB_WRITE_5_CMD
```

```
0000078E' 15B7  2919                   .ADDRESS        B_DATA_PKT_3
0000078E' 15BB  2920                   .ADDRESS        B_DATA_PKT_3
0000078E' 15BF  2921                   .ADDRESS        B_DATA_PKT_3
0000078E' 15C3  2922                   .ADDRESS        B_DATA_PKT_3
0000068A' 15C7  2923                   .ADDRESS        MB_END_PACKET_4
          15CB  2924
000003A4' 15CB  2925                   .ADDRESS        MB_WRITE_6_CMD
0000078E' 15CF  2926                   .ADDRESS        B_DATA_PKT_3
0000078E' 15D3  2927                   .ADDRESS        B_DATA_PKT_3
0000078E' 15D7  2928                   .ADDRESS        B_DATA_PKT_3
0000078E' 15DB  2929                   .ADDRESS        B_DATA_PKT_3
0000068A' 15DF  2930                   .ADDRESS        MB_END_PACKET_4
          15E3  2931
000003B2' 15E3  2932                   .ADDRESS        MB_WRITE_7_CMD
00000812' 15E7  2933                   .ADDRESS        B_DATA_PKT_4
000006A6' 15EB  2934                   .ADDRESS        MB_END_PACKET_6
          15EF  2935
000003C0' 15EF  2936                   .ADDRESS        MB_READ_5_CMD
00000817' 15F3  2937                   .ADDRESS        B_DATA_PKT_5
0000089B' 15F7  2938                   .ADDRESS        B_DATA_PKT_6
0000089B' 15FB  2939                   .ADDRESS        B_DATA_PKT_6
0000089B' 15FF  2940                   .ADDRESS        B_DATA_PKT_6
0000068A' 1603  2941                   .ADDRESS        MB_END_PACKET_4
          1607  2942
000003CE' 1607  2943                   .ADDRESS        MB_READ_6_CMD
0000078E' 160B  2944                   .ADDRESS        B_DATA_PKT_3
0000078E' 160F  2945                   .ADDRESS        B_DATA_PKT_3
0000078E' 1613  2946                   .ADDRESS        B_DATA_PKT_3
0000078E' 1617  2947                   .ADDRESS        B_DATA_PKT_3
0000068A' 161B  2948                   .ADDRESS        MB_END_PACKET_4
          161F  2949
          161F  2950 ; This is the packet list for the 'write record padded zeros  subtest.
          161F  2951
000003DC' 161F  2952 MCOMMAND_TBL_2:   .ADDRESS        MB_WRITE_8_CMD
0000078E' 1623  2953                   .ADDRESS        B_DATA_PKT_3
00000698' 1627  2954                   .ADDRESS        MB_END_PACKET_5
          162B  2955
000003EA' 162B  2956                   .ADDRESS        MB_WRITE_9_CMD
0000078E' 162F  2957                   .ADDRESS        B_DATA_PKT_3
00000698' 1633  2958                   .ADDRESS        MB_END_PACKET_5
          1637  2959
000003F8' 1637  2960                   .ADDRESS        MB_WRITE_10_CMD
00000812' 163B  2961                   .ADDRESS        B_DATA_PKT_4
000006A6' 163F  2962                   .ADDRESS        MB_END_PACKET_6
          1643  2963
00000406' 1643  2964                   .ADDRESS        MB_READ_7_CMD
00000817' 1647  2965                   .ADDRESS        B_DATA_PKT_5
00000698' 164B  2966                   .ADDRESS        MB_END_PACKET_5
          164F  2967
00000414' 164F  2968                   .ADDRESS        MB_READ_8_CMD
0000078E' 1653  2969                   .ADDRESS        B_DATA_PKT_3
00000698' 1657  2970                   .ADDRESS        MB_END_PACKET_5
          165B  2971
          165B  2972 ; This is the packet list for the 'block write read  subtest.
          165B  2973
```

```
00000422' 165B  2974 MCOMMAND_TBL_3: .ADDRESS         MB_WRITE_11_CMD
0000089B' 165F  2975                 .ADDRESS         B_DATA_PKT_6
0000089B' 1663  2976                 .ADDRESS         B_DATA_PKT_6
0000089B' 1667  2977                 .ADDRESS         B_DATA_PKT_6
0000089B' 166B  2978                 .ADDRESS         B_DATA_PKT_6
0000091F' 166F  2979                 .ADDRESS         B_DATA_PKT_7
0000091F' 1673  2980                 .ADDRESS         B_DATA_PKT_7
0000091F' 1677  2981                 .ADDRESS         B_DATA_PKT_7
0000091F' 167B  2982                 .ADDRESS         B_DATA_PKT_7
000006B4' 167F  2983                 .ADDRESS         MB_END_PACKET_7
          1683  2984
00000430' 1683  2985                 .ADDRESS         MB_READ_9_CMD
0000089B' 1687  2986                 .ADDRESS         B_DATA_PKT_6
0000089B' 168B  2987                 .ADDRESS         B_DATA_PKT_6
0000089B' 168F  2988                 .ADDRESS         B_DATA_PKT_6
0000089B' 1693  2989                 .ADDRESS         B_DATA_PKT_6
0000091F' 1697  2990                 .ADDRESS         B_DATA_PKT_7
0000091F' 169B  2991                 .ADDRESS         B_DATA_PKT_7
0000091F' 169F  2992                 .ADDRESS         B_DATA_PKT_7
0000091F' 16A3  2993                 .ADDRESS         B_DATA_PKT_7
000006B4' 16A7  2994                 .ADDRESS         MB_END_PACKET_7
          16AB  2995
0000043E' 16AB  2996                 .ADDRESS         MB_WRITE_12_CMD
000009A3' 16AF  2997                 .ADDRESS         B_DATA_PKT_8
000009A3' 16B3  2998                 .ADDRESS         B_DATA_PKT_8
000009A3' 16B7  2999                 .ADDRESS         B_DATA_PKT_8
000009A3' 16BB  3000                 .ADDRESS         B_DATA_PKT_8
0000068A' 16BF  3001                 .ADDRESS         MB_END_PACKET_4
          16C3  3002
0000044C' 16C3  3003                 .ADDRESS         MB_READ_10_CMD
000009A3' 16C7  3004                 .ADDRESS         B_DATA_PKT_8
000009A3' 16CB  3005                 .ADDRESS         B_DATA_PKT_8
000009A3' 16CF  3006                 .ADDRESS         B_DATA_PKT_8
000009A3' 16D3  3007                 .ADDRESS         B_DATA_PKT_8
0000068A' 16D7  3008                 .ADDRESS         MB_END_PACKET_4
          16DB  3009
0000045A' 16DB  3010                 .ADDRESS         MB_WRITE_13_CMD
00000A27' 16DF  3011                 .ADDRESS         B_DATA_PKT_9
00000A27' 16E3  3012                 .ADDRESS         B_DATA_PKT_9
00000A27' 16E7  3013                 .ADDRESS         B_DATA_PKT_9
00000A27' 16EB  3014                 .ADDRESS         B_DATA_PKT_9
0000068A' 16EF  3015                 .ADDRESS         MB_END_PACKET_4
          16F3  3016
00000468' 16F3  3017                 .ADDRESS         MB_READ_11_CMD
00000A27' 16F7  3018                 .ADDRESS         B_DATA_PKT_9
00000A27' 16FB  3019                 .ADDRESS         B_DATA_PKT_9
00000A27' 16FF  3020                 .ADDRESS         B_DATA_PKT_9
00000A27' 1703  3021                 .ADDRESS         B_DATA_PKT_9
0000068A' 1707  3022                 .ADDRESS         MB_END_PACKET_4
          170B  3023
00000492' 170B  3024                 .ADDRESS         MB_WRITE_15_CMD
00000C2B' 170F  3025                 .ADDRESS         B_DATA_PKT_13
00000C2B' 1713  3026                 .ADDRESS         B_DATA_PKT_13
00000C2B' 1717  3027                 .ADDRESS         B_DATA_PKT_13
00000C2B' 171B  3028                 .ADDRESS         B_DATA_PKT_13
```

```
0000068A' 171F  3029                .ADDRESS        MB_END_PACKET_4
          1723  3030
000004A0' 1723  3031                .ADDRESS        MB_READ_13_CMD
00000C2B' 1727  3032                .ADDRESS        B_DATA_PKT_13
00000C2B' 172B  3033                .ADDRESS        B_DATA_PKT_13
00000C2B' 172F  3034                .ADDRESS        B_DATA_PKT_13
00000C2B' 1733  3035                .ADDRESS        B_DATA_PKT_13
0000068A' 1737  3036                .ADDRESS        MB_END_PACKET_4
          173B  3037
000004AE' 173B  3038                .ADDRESS        MB_WRITE_16_CMD
00000CAF' 173F  3039                .ADDRESS        B_DATA_PKT_14
00000CAF' 1743  3040                .ADDRESS        B_DATA_PKT_14
00000CAF' 1747  3041                .ADDRESS        B_DATA_PKT_14
00000CAF' 174B  3042                .ADDRESS        B_DATA_PKT_14
0000068A' 174F  3043                .ADDRESS        MB_END_PACKET_4
          1753  3044
000004BC' 1753  3045                .ADDRESS        MB_READ_14_CMD
00000CAF' 1757  3046                .ADDRESS        B_DATA_PKT_14
00000CAF' 175B  3047                .ADDRESS        B_DATA_PKT_14
00000CAF' 175F  3048                .ADDRESS        B_DATA_PKT_14
00000CAF' 1763  3049                .ADDRESS        B_DATA_PKT_14
0000068A' 1767  3050                .ADDRESS        MB_END_PACKET_4
          176B  3051
          176B  3052 ; This is the packet list for the 'record write read  subtest
          176B  3053
000004E6' 176B  3054 MCOMMAND_TBL_4: .ADDRESS       MB_WRITE_18_CMD
00000EBF' 176F  3055                .ADDRESS        B_DATA_PKT_18
00000EBF' 1773  3056                .ADDRESS        B_DATA_PKT_18
000006D0' 1777  3057                .ADDRESS        MB_END_PACKET_9
          177B  3058
000004F4' 177B  3059                .ADDRESS        MB_READ_16_CMD
00000EBF' 177F  3060                .ADDRESS        B_DATA_PKT_18
00000EBF' 1783  3061                .ADDRESS        B_DATA_PKT_18
000006D0' 1787  3062                .ADDRESS        MB_END_PACKET_9
          178B  3063
00000502' 178B  3064                .ADDRESS        MB_WRITE_19_CMD
00000F43' 178F  3065                .ADDRESS        B_DATA_PKT_19
00000698' 1793  3066                .ADDRESS        MB_END_PACKET_5
          1797  3067
00000510' 1797  3068                .ADDRESS        MB_READ_17_CMD
00000F43' 179B  3069                .ADDRESS        B_DATA_PKT_19
00000698' 179F  3070                .ADDRESS        MB_END_PACKET_5
          17A3  3071
0000051E' 17A3  3072                .ADDRESS        MB_WRITE_20_CMD
00000FC7' 17A7  3073                .ADDRESS        B_DATA_PKT_20
00000698' 17AB  3074                .ADDRESS        MB_END_PACKET_5
          17AF  3075
0000052C' 17AF  3076                .ADDRESS        MB_READ_18_CMD
00000FC7' 17B3  3077                .ADDRESS        B_DATA_PKT_20
00000698' 17B7  3078                .ADDRESS        MB_END_PACKET_5
          17BB  3079
00000556' 17BB  3080                .ADDRESS        MB_WRITE_22_CMD
00001137' 17BF  3081                .ADDRESS        B_DATA_PKT_23
00000698' 17C3  3082                .ADDRESS        MB_END_PACKET_5
          17C7  3083
```

```
00000564' 17C7  3084                    .ADDRESS        MB_READ_20_CMD
00001137' 17CB  3085                    .ADDRESS        B_DATA_PKT_23
00000698' 17CF  3086                    .ADDRESS        MB_END_PACKET_5
          17D3  3087
00000572' 17D3  3088                    .ADDRESS        MB_WRITE_23_CMD
000011BB' 17D7  3089                    .ADDRESS        B_DATA_PKT_24
00000698' 17DB  3090                    .ADDRESS        MB_END_PACKET_5
          17DF  3091
00000580' 17DF  3092                    .ADDRESS        MB_READ_21_CMD
000011BB' 17E3  3093                    .ADDRESS        B_DATA_PKT_24
00000698' 17E7  3094                    .ADDRESS        MB_END_PACKET_5
          17EB  3095
          17EB  3096
000005AA' 17EB  3097 MCOMMAND_TBL_5:  .ADDRESS        MB_READ_23_CMD
0000089B' 17EF  3098                    .ADDRESS        B_DATA_PKT_6
0000089B' 17F3  3099                    .ADDRESS        B_DATA_PKT_6
0000089B' 17F7  3100                    .ADDRESS        B_DATA_PKT_6
0000089B' 17FB  3101                    .ADDRESS        B_DATA_PKT_6
0000068A' 17FF  3102                    .ADDRESS        MB_END_PACKET_4
          1803  3103
000005B8' 1803  3104                    .ADDRESS        MB_READ_24_CMD
00000C2B' 1807  3105                    .ADDRESS        B_DATA_PKT_13
00000C2B' 180B  3106                    .ADDRESS        B_DATA_PKT_13
00000C2B' 180F  3107                    .ADDRESS        B_DATA_PKT_13
00000C2B' 1813  3108                    .ADDRESS        B_DATA_PKT_13
0000068A' 1817  3109                    .ADDRESS        MB_END_PACKET_4
          181B  3110
000005C6' 181B  3111                    .ADDRESS        MB_POS_1_CMD
00000660' 181F  3112                    .ADDRESS        MB_END_PACKET_1
          1823  3113
000005AA' 1823  3114                    .ADDRESS        MB_READ_23_CMD
0000089B' 1827  3115                    .ADDRESS        B_DATA_PKT_6
0000089B' 182B  3116                    .ADDRESS        B_DATA_PKT_6
0000089B' 182F  3117                    .ADDRESS        B_DATA_PKT_6
0000089B' 1833  3118                    .ADDRESS        B_DATA_PKT_6
0000068A' 1837  3119                    .ADDRESS        MB_END_PACKET_4
          183B  3120
```

```
                                     183B  3122 .SBTTL ERROR MESSAGES
                                     183B  3123 ;------------------------------------------------------------------------
                                     183B  3124 T_INIT1_ERR:
20 67 6E 69 74 70 6D 65 74 74 61 00' 183B  3125          .ASCIC 'attempting to initialize TU58 CONTROLLER.
7A 69 6C 61 69 74 69 6E 69 20 6F 74  1847
52 54 4E 4F 43 20 38 35 55 54 20 65  1853
                  2E 52 45 4C 4C 4F  185F
                                 29  183B
                                     1865  3126 ;------------------------------------------------------------------------
                                     1865  3127 T_INIT2_ERR:
20 67 6E 69 74 70 6D 65 74 74 61 00' 1865  3128          .ASCIC  attempting an INITIALIZE function.
5A 49 4C 41 49 54 49 4E 49 20 6E 61  1871
   2E 6E 6F 69 74 63 6E 75 66 20 45  187D
                                 22  1865
                                     1888  3129 ;------------------------------------------------------------------------
                                     1888  3130 T_SLF_DIAG_ERR:
20 67 6E 69 74 70 6D 65 74 74 61 00' 1888  3131          .ASCIC  attempting a SELF-DIAGNOSE function."
4E 47 41 49 44 2D 46 4C 45 53 20 61  1894
6E 6F 69 74 63 6E 75 66 20 45 53 4F  18A0
                                 2E  18AC
                                 24  1888
                                     18AD  3132 ;------------------------------------------------------------------------
                                     18AD  3133 T_NOP_ERR:
20 67 6E 69 74 70 6D 65 74 74 61 00' 18AD  3134          .ASCIC  attempting a NOP function.
69 74 63 6E 75 66 20 50 4F 4E 20 61  18B9
                           2E 6E 6F  18C5
                                 1A  18AD
                                     18C8  3135 ;------------------------------------------------------------------------
                                     18C8  3136 T_WR_BLK_ERR:
20 67 6E 69 74 70 6D 65 74 74 61 00' 18C8  3137          .ASCIC 'attempting a WRITE BLOCK function.'
43 4F 4C 42 20 45 54 49 52 57 20 61  18D4
   2E 6E 6F 69 74 63 6E 75 66 20 4B  18E0
                                 22  18C8
                                     18EB  3138 ;------------------------------------------------------------------------
                                     18EB  3139 T_RD_BLK_ERR:
20 67 6E 69 74 70 6D 65 74 74 61 00' 18EB  3140          .ASCIC  attempting a READ BLOCK function.'
4B 43 4F 4C 42 20 44 41 45 52 20 61  18F7
      2E 6E 6F 69 74 63 6E 75 66 20  1903
                                 21  18EB
                                     190D  3141 ;------------------------------------------------------------------------
                                     190D  3142 T_WR_REC_ERR:
20 67 6E 69 74 70 6D 65 74 74 61 00' 190D  3143          .ASCIC  attempting a WRITE RECORD function.'
4F 43 45 52 20 45 54 49 52 57 20 61  1919
2E 6E 6F 69 74 63 6E 75 66 20 44 52  1925
                                 23  190D
                                     1931  3144 ;------------------------------------------------------------------------
                                     1931  3145 T_RD_REC_ERR:
20 67 6E 69 74 70 6D 65 74 74 61 00' 1931  3146          .ASCIC 'attempting a READ RECORD function.'
52 4F 43 45 52 20 44 41 45 52 20 61  193D
   2E 6E 6F 69 74 63 6E 75 66 20 44  1949
                                 22  1931
                                     1954  3147 ;-------------- ---------------------------------------------------------
                                     1954  3148 T_WR_MOD_ERR:
20 67 6E 69 74 70 6D 65 74 74 61 00' 1954  3149          .ASCIC  attempting a WRITE MODIFIED function.
49 44 4F 4D 20 45 54 49 52 57 20 61  1960
```

```
6F 69 74 63 6E 75 66 20 44 45 49 46  196C
                              2E 6E  1978
                                 25  1954
                                     197A   3150 ;---------------------------------------------------------------------
                                     197A   3151 T_RD_MOD_ERR:
20 67 6E 69 74 70 6D 65 74 74 61 00' 197A   3152         .ASCIC "attempting a READ MODIFIED function.'
46 49 44 4F 4D 20 44 41 45 52 20 61  1986
6E 6F 69 74 63 6E 75 66 20 44 45 49  1992
                                 2E  199E
                                 24  197A
                                     199F   3153 ;---------------------------------------------------------------------
                                     199F   3154 T_RD_INT_ERR:
20 67 6E 69 74 70 6D 65 74 74 61 00' 199F   3155         .ASCIC  attempting a READ RECORD INTERRUPT-MODE function."
52 4F 43 45 52 20 44 41 45 52 20 61  19AB
2D 54 50 55 52 52 45 54 4E 49 20 44  19B7
6F 69 74 63 6E 75 66 20 45 44 4F 4D  19C3
                              2E 6E  19CF
                                 31  199F
                                     19D1   3156 ;---------------------------------------------------------------------
                                     19D1   3157 T_WR_INT_ERR:
20 67 6E 69 74 70 6D 65 74 74 61 00' 19D1   3158         .ASCIC  attempting a WRITE RECORD INTERRUPT-MODE function."
4F 43 45 52 20 45 54 49 52 57 20 61  19DD
54 50 55 52 52 45 54 4E 49 20 44 52  19E9
69 74 63 6E 75 66 20 45 44 4F 4D 2D  19F5
                           2E 6E 6F  1A01
                                 32  19D1
                                     1A04   3159 ;---------------------------------------------------------------------
                                     1A04   3160 T_IPL_ERR:
20 67 6E 69 74 70 6D 65 74 74 61 00' 1A04   3161         .ASCIC  attempting a NOP function to check IPL.'
69 74 63 6E 75 66 20 50 4F 4E 20 61  1A10
20 6B 63 65 68 63 20 6F 74 20 6E 6F  1A1C
                        2E 4C 50 49  1A28
                                 27  1A04
                                     1A2C   3162 ;---------------------------------------------------------------------
                                     1A2C   3163 T_INT_ERR:
74 70 75 72 72 65 74 6E 49 20 20 00' 1A2C   3164         .ASCIC    Interrupt occurred when IPL should have prohibited it."
68 77 20 64 65 72 72 75 63 63 6F 20  1A38
6C 75 6F 68 73 20 4C 50 49 20 6E 65  1A44
69 68 6F 72 70 20 65 76 61 68 20 64  1A50
         2E 74 69 20 64 65 74 69 62  1A5C
                                 38  1A2C
                                     1A65   3165 ;---------------------------------------------------------------------
                                     1A65   3166 T_NO_INT_ERR:
20 64 65 74 63 65 70 78 45 20 20 00' 1A65   3167         .ASCIC '  Expected interrupt did not occur.'
69 64 20 74 70 75 72 72 65 74 6E 69  1A71
2E 72 75 63 63 6F 20 74 6F 6E 20 64  1A7D
                                 23  1A65
                                     1A89   3168 ;---------------------------------------------------------------------
                                     1A89   3169 T_SEND_INT_ERR:
20 64 65 74 63 65 70 78 45 20 20 00' 1A89   3170         .ASCIC    Expected INTERRUPT did not occur while sending DATA packet.!/
69 64 20 54 50 55 52 52 45 54 4E 49  1A95
20 72 75 63 63 6F 20 74 6F 6E 20 64  1AA1
6E 69 64 6E 65 73 20 65 6C 69 68 77  1AAD
65 6B 63 61 70 20 41 54 41 44 20 67  1AB9
                        2F 21 2E 74  1AC5
```

```
                                3F  1A89
                                    1AC9   3171 SEND_INT_ERR:
                                    1AC9   3172            $DS_BGNMESSAGE
                                    1ACB   3173            $DS_PRINTF_S    FORMAT=T_SEND_INT_ERR
                                    1AD5   3174            $DS_ENDMESSAGE
                                    1AD6   3175 ;-------------------------------------------------------------------
                                    1AD6   3176 MSG_1:
                                    1AD6   3177            $DS_BGNMESSAGE
                                    1AD8   3178            $DS_PRINTF_S    FORMAT=T_MSG_1
                                    1AE5   3179            $DS_ENDMESSAGE
                                    1AE6   3180 ;-------------------------------------------------------------------
                                    1AE6   3181 MSG_2:
                                    1AE6   3182            $DS_BGNMESSAGE
                                    1AE8   3183            $DS_PRINTF_S    FORMAT=T_MSG_2
                                    1AF5   3184            $DS_ENDMESSAGE
                                    1AF6   3185 ;-------------------------------------------------------------------
                                    1AF6   3186 T_MSG_1:
61 68 74 20 6B 63 65 68 43 2F 21 00' 1AF6   3187            .ASCIC  '!/Check that a test tape is loaded in the drive!/
70 61 74 20 74 73 65 74 20 61 20 74  1B02
20 64 65 64 61 6F 6C 20 73 69 20 65  1B0E
65 76 69 72 64 20 65 68 74 20 6E 69  1B1A
                             2F 21  1B26
                                31  1AF6
                                    1B28   3188 ;-------------------------------------------------------------------
                                    1B28   3189 T_MSG_2:
61 68 74 20 6B 63 65 68 43 2F 21 00' 1B28   3190            .ASCIC  '!/Check that the test tape is write enabled!/
74 20 74 73 65 74 20 65 68 74 20 74  1B34
65 74 69 72 77 20 73 69 20 65 70 61  1B40
      2F 21 64 65 6C 62 61 6E 65 20  1B4C
                                2D  1B28
                                    1B56   3191 ;-------------------------------------------------------------------
                                    1B56   3192 T_OVERWRITE:
20 38 35 55 54 20 65 68 54 0A 0D 00' 1B56   3193            .ASCIC  <13><10> The TU58 tape cartridge is not recognized as a"<13><10>-
64 69 72 74 72 61 63 20 65 70 61 74  1B62
65 72 20 74 6F 6E 20 73 69 20 65 67  1B6E
20 73 61 20 64 65 7A 69 6E 67 6F 63  1B7A
74 20 68 63 74 61 72 63 73 0A 0D 61  1B86
20 75 6F 79 20 6F 64 20 2C 65 70 61  1B92
72 65 76 6F 20 6F 74 20 68 73 69 77  1B9E
61 74 20 65 68 74 20 65 74 69 72 77  1BAA
                          3F 65 70  1BB6
                                62  1B56
                                    1BB9   3194                          scratch tape, do you wish to overwrite the tape?
                                    1BB9   3195 ;-------------------------------------------------------------------
                                    1BB9   3196 T_NO_READY:
64 69 64 20 79 64 61 65 52 20 20 00' 1BB9   3197            .ASCIC     Ready did not set after sending a byte to TU58.
74 66 61 20 74 65 73 20 74 6F 6E 20  1BC5
61 20 67 6E 69 64 6E 65 73 20 72 65  1BD1
35 55 54 20 6F 74 20 65 74 79 62 20  1BDD
                             2E 38  1BE9
                                31  1BB9
                                    1BEB   3198 NO_READY:
                                    1BEB   3199            $DS_BGNMESSAGE
                                    1BED   3200            $DS_PRINTX_S    FORMAT = T_NO_READY
                                    1BF7   3201            $DS_ENDMESSAGE
```

```
                                     1BF8   3202 ;----------------------------------------------------------------------------
                                     1BF8   3203 T_BAD_CONT:
20 61 74 61 64 20 64 61 42 20 20 00' 1BF8   3204         .ASCIC  '   Bad data in CONTINUE flag.!/!/'
20 45 55 4E 49 54 4E 4F 43 20 6E 69  1C04
      2F 21 2F 21 2E 67 61 6C 66     1C10
                                 20  1BF8
                                     1C19   3205 ;----------------------------------------------------------------------------
                                     1C19   3206 T_BAD_END:
20 61 74 61 64 20 64 61 42 20 20 00' 1C19   3207         .ASCIC  '   Bad data in END packet.!/!/'
65 6B 63 61 70 20 44 4E 45 20 6E 69  1C25
                  2F 21 2F 21 2E 74  1C31
                                 1D  1C19
                                     1C37   3208 ;----------------------------------------------------------------------------
                                     1C37   3209 T_BAD_DATA:
20 61 74 61 64 20 64 61 42 20 20 00' 1C37   3210         .ASCIC      Bad data in DATA packet.!/!/'
6B 63 61 70 20 41 54 41 44 20 6E 69  1C43
               2F 21 2F 21 2E 74 65  1C4F
                                 1E  1C37
                                     1C56   3211 ;----------------------------------------------------------------------------
                                     1C56   3212 T_EXP_CONT:
3A 44 45 54 43 45 50 58 45 20 20 00' 1C56   3213         .ASCIC      EXPECTED:    !XB!/'
                  2F 21 42 58 21 09  1C62
                                 11  1C56
                                     1C68   3214 ;----------------------------------------------------------------------------
                                     1C68   3215 T_ACT_CONT:
20 20 3A 4C 41 55 54 43 41 20 20 00' 1C68   3216         .ASCIC  '   ACTUAL:      !XB!/
                  2F 21 42 58 21 09  1C74
                                 11  1C68
                                     1C7A   3217 ;----------------------------------------------------------------------------
                                     1C7A   3218 T_EXP_DATA:
20 44 45 54 43 45 50 58 45 20 20 00' 1C7A   3219         .ASCIC  '   EXPECTED PACKET ADDRESS:     !XL!/'
45 52 44 44 41 20 54 45 4B 43 41 50  1C86
      2F 21 4C 58 21 09 3A 53 53     1C92
                                 20  1C7A
                                     1C9B   3220 ;----------------------------------------------------------------------------
                                     1C9B   3221 T_ACT_DATA:
41 50 20 4C 41 55 54 43 41 20 20 00' 1C9B   3222         .ASCIC      ACTUAL PACKET ADDRESS:       !XL!/
53 53 45 52 44 44 41 20 54 45 4B 43  1CA7
   2F 21 4C 58 21 09 20 20 20 20 3A  1CB3
                                 22  1C9B
                                     1CBE   3223 ;----------------------------------------------------------------------------
                                     1CBE   3224 T_DUMP_HDR:
3A 44 45 54 43 45 50 58 45 20 20 00' 1CBE   3225         .ASCIC      EXPECTED:    ACTUAL:         XOR:!/"
4F 58 09 09 3A 4C 41 55 54 43 41 09  1CCA
                        2F 21 3A 52  1CD6
                                 1B  1CBE
                                     1CDA   3226 ;----------------------------------------------------------------------------
                                     1CDA   3227 T_DUMP_XOR:
20 09 09 42 58 21 20 20 20 20 20 00' 1CDA   3228         .ASCIC        !XB                !XB             !XB!/
2F 21 42 58 21 20 09 09 42 58 21 20  1CE6
                                 17  1CDA
                                     1CF2   3229 ;----------------------------------------------------------------------------
                                     1CF2   3230 T_DUMP_NOXOR:
20 09 09 42 58 21 20 20 20 20 20 00' 1CF2   3231         .ASCIC        !XB                !XB!/"
                  2F 21 42 58 21 20  1CFE
```

```
                                 11  1CF2
                                     1D04   3232 ;-----------------------------------------------------------
                                     1D04   3233 T_REC_ERR:
74 63 65 72 72 6F 63 6E 49 20 20 00' 1D04   3234            .ASCIC  '  Incorrect number of bytes received.!/!/
62 20 66 6F 20 72 65 62 6D 75 6E 20  1D10
65 76 69 65 63 65 72 20 73 65 74 79  1D1C
                  2F 21 2F 21 2E 64  1D28
                                 29  1D04
                                     1D2E   3235 ;---------------------------- ------------------------------
                                     1D2E   3236 T_REC_END:
3A 44 45 54 43 45 50 58 45 20 20 00' 1D2E   3237            .ASCIC  "  EXPECTED: END packet!/
21 74 65 6B 63 61 70 20 44 4E 45 20  1D3A
                                 2F  1D46
                                 18  1D2E
                                     1D47   3238 ;-----------------------------------------------------------
                                     1D47   3239 T_REC_CONT:
3A 44 45 54 43 45 50 58 45 20 20 00' 1D47   3240            .ASCIC     EXPECTED: CONTINUE flag = 01!/'
6C 66 20 45 55 4E 49 54 4E 4F 43 20  1D53
         2F 21 31 30 20 3D 20 67 61  1D5F
                                 20  1D47
                                     1D68   3241 ;-----------------------------------------------------------
                                     1D68   3242 T_REC_DATA:
3A 44 45 54 43 45 50 58 45 20 20 00' 1D68   3243            .ASCIC  '  EXPECTED: DATA packet beginning at !XL!/
74 65 6B 63 61 70 20 41 54 41 44 20  1D74
61 20 67 6E 69 6E 6E 69 67 65 62 20  1D80
               2F 21 4C 58 21 20 74  1D8C
                                 2A  1D68
                                     1D93   3244 ;-----------------------------------------------------------
                                     1D93   3245 T_EMPTY_BUF:
72 65 77 20 65 72 65 68 54 20 20 00' 1D93   3246            .ASCIC     There were no bytes received.!/'
72 20 73 65 74 79 62 20 6F 6E 20 65  1D9F
      2F 21 2E 64 65 76 69 65 63 65  1DAB
                                 21  1D93
                                     1DB5   3247 ;-----------------------------------------------------------
                                     1DB5   3248 T_ACT_END:
20 74 73 61 6C 20 65 68 54 20 20 00' 1DB5   3249            .ASCIC     The last E(X) bytes received were the following END packet!/
72 20 73 65 74 79 62 20 29 58 28 45  1DC1
65 72 65 77 20 64 65 76 69 65 63 65  1DCD
69 77 6F 6C 6C 6F 66 20 65 68 74 20  1DD9
65 6B 63 61 70 20 44 4E 45 20 67 6E  1DE5
                           2F 21 74  1DF1
                                 3E  1DB5
                                     1DF4   3250 ;-----------------------------------------------------------
                                     1DF4   3251 T_UNREC_DATA:
69 6E 67 6F 63 65 72 6E 55 20 20 00' 1DF4   3252            .ASCIC  '  Unrecognizable packet received.!/
74 65 6B 63 61 70 20 65 6C 62 61 7A  1E00
2F 21 2E 64 65 76 69 65 63 65 72 20  1E0C
                                 23  1DF4
                                     1E18   3253 ;-----------------------------------------------------------
                                     1E18   3254 T_ACT_REC:
21 20 3A 4C 41 55 54 43 41 20 20 00' 1E18   3255            .ASCIC     ACTUAL: !XL(X) bytes beginning at !XL!/
20 73 65 74 79 62 20 29 58 28 4C 58  1E24
74 61 20 67 6E 69 6E 6E 69 67 65 62  1E30
               2F 21 4C 58 21 20     1E3C
                                 29  1E18
```

```
                                     1E42  3256 ;---------------------------------------------------------------------------
                                     1E42  3257 T_DUMP_14:
   2F 21 42 58 21 20 20 20 20 20 00' 1E42  3258         .ASCIC  "      !XB!/'
                                 0A  1E42
                                     1E4D  3259 ;---------------------------------------------------------------------------
                                     1E4D  3260 DATA_ERR_CONT:
                                     1E4D  3261         $DS_BGNMESSAGE
                                     1E4F  3262         $DS_PRINTX_S    FORMAT = T_BAD_CONT
                                     1E5A  3263         $DS_PRINTX_S    FORMAT = T_EXP_CONT, P1 = (R10)
                                     1E67  3264         $DS_PRINTX_S    FORMAT = T_ACT_CONT, P1 = (R11)
                                     1E74  3265         $DS_ENDMESSAGE
                                     1E75  3266 ;---------------------------------------------------------------------------
                                     1E75  3267 DATA_ERR_DATA:
                                     1E75  3268         $DS_BGNMESSAGE
                                     1E77  3269         $DS_PRINTX_S    FORMAT = T_BAD_DATA
                                     1E82  3270         $DS_PRINTX_S    FORMAT = T_EXP_DATA, P1 = R10
                                     1E8F  3271         $DS_PRINTX_S    FORMAT = T_ACT_DATA, P1 = R11
                                     1E9C  3272         $DS_ENDMESSAGE
                                     1E9D  3273 ;---------------------------------------------------------------------------
                                     1E9D  3274 DATA_ERR_END:
                                     1E9D  3275         $DS_BGNMESSAGE
                                     1E9F  3276         $DS_PRINTX_S    FORMAT = T_BAD_END
                       5C   0E   D0  1EAA  3277         MOVL    #14,R12
                   000021E7'EF   16  1EAD  3278         JSB     DUMP_PKT
                                     1EB3  3279         $DS_ENDMESSAGE
                                     1EB4  3280 ;---------------------------------------------------------------------------
                                     1EB4  3281 ; These GPR's are used in the following error-reporting routines:
                                     1EB4  3282
                                     1EB4  3283 ; R5 contains the next address in the receive buffer to be loaded
                                     1EB4  3284 ; R10 contains the address of the beginning of the expected data buffer
                                     1EB4  3285 ; R11 contains the address of the beginning of the actual data buffer
                                     1EB4  3286
                                     1EB4  3287
                                     1EB4  3288 REC_ERR:
                                     1EB4  3289         $DS_BGNMESSAGE
                                     1EB6  3290         $DS_PRINTX_S    FORMAT=T_REC_ERR ; print extended message
               0C E13A CF   02   E0  1EC1  3291         BBS     #V_DATA_FLAG,B_FLAGS,20$ ; br = data packet expected
                                     1EC7  3292         $DS_PRINTX_S    FORMAT=@ERR$_P1(AP) ; identify expected packet as
                            0C   11  1ED1  3293         BRB     30$                     ;             end or continue
                                     1ED3  3294 20$:    $DS_PRINTX_S    FORMAT=@ERR$_P1(AP),P1=R10 ; ident. exp. data packet
               09 E11C CF   03   E1  1EDF  3295 30$:    BBC     #V_END_FLAG,B_FLAGS,50$ ; br = end not expected
                       59   5A   D0  1EE5  3296         MOVL    R10,R9                  ; load expected end packet address
                   00002227'EF   16  1EE8  3297         JSB     DUMP_14                 ; go print expected end packet
                  57   55   5B   C3  1EEE  3298 50$:    SUBL3   R11,R5,R7               ; load actual byte count
                            0D   12  1EF2  3299         BNEQ    60$                     ; br = buffer not empty
                                     1EF4  3300         $DS_PRINTX_S FORMAT=T_EMPTY_BUF ; print empty buffer message
                            3B   11  1EFF  3301         BRB     100$                    ; done if empty buffer
                       0E   57   D1  1F01  3302 60$:    CMPL    R7,#14                  ; check byte count
                            1C   19  1F04  3303         BLSS    70$                     ; br = less than 14 bytes received
                   0000223D'EF   16  1F06  3304         JSB     FIND_END                ; go check for unexpected end packet
                                     1F0C  3305         $DS_BNCOMPLETE  70$             ; br = end not found
                                     1F0F  3306         $DS_PRINTX_S    FORMAT=T_ACT_END ; print end packet found message
                   00002227'EF   16  1F1A  3307         JSB     DUMP_14                 ; print end packet received
                            0B   11  1F20  3308         BRB     80$                     ; done
                                     1F22  3309 70$:    $DS_PRINTX_S    FORMAT=T_UNREC_DATA ; identify as unknown data
```

```
1F2D  3310 80$:    $DS_PRINTX_S    FORMAT=T_ACT_REC,P1=R7,P2=R11 ; print byte count and
1F3C  3311 100$:   $DS_ENDMESSAGE                                ;       buffer address
1F3D  3312
1F3D  3313 ;-----------------------------------------------------------------------------
1F3D  3314 INT_ERR:
1F3D  3315         $DS_BGNMESSAGE
1F3F  3316         $DS_PRINTX_S    FORMAT=T_INT_ERR
1F4A  3317         $DS_ENDMESSAGE
1F4B  3318 ;------------------------------------------------------------------------------
1F4B  3319 NO_INT_ERR:
1F4B  3320         $DS_BGNMESSAGE
1F4D  3321         $DS_PRINTX_S    FORMAT=T_NO_INT_ERR
1F58  3322         $DS_ENDMESSAGE
1F59  3323 ;-----------------------------------------------------------------------------
1F59  3324
1F59  3325
1F59  3326
1F59  3327
1F59  3328
1F59  3332
```

```
                                       1F59   3334 .SBTTL  SUBROUTINES
                                       1F59   3335
                                       1F59   3336 ; This subroutine will do an init sequence to the TU58. If the TU58
                                       1F59   3337 ; does not respond or does not send a continue, an error will be
                                       1F59   3338 ; reported in the main line code. This routine is called at the
                                       1F59   3339 ; beginning of each TU58 test. When calling INIT_SEQUENCE R4 must
                                       1F59   3340 ; contain the delay time for sending a break.
                                       1F59   3341
                                       1F59   3342 INIT_SEQUENCE:
                  E0A3 CF       94     1F59   3343          CLRB    B_FLAGS                    ; initialize timeout flag
             54   FFF85EE0 8F   D0     1F5D   3344          MOVL    #DELAY_TIME2,R4            ; load delay time
                    57    02    D0     1F64   3345          MOVL    #2,R7                      ; load number of bytes to send
               55   E0B6  CF    9E     1F67   3346          MOVAB   B_ZEROS,R5                 ; load address of data
                    1E    01    DA     1F6C   3347          MTPR    #1,#PR750$_CSTS            ; set break bit          5.2
                0000203D'EF     16     1F6F   3348          JSB     SEND_TU58                  ; send zeros
              4A E086 CF    00  E0     1F75   3349 1$:      BBS     #V_CLOCK_FLAG,B_FLAGS,10$ ; exit if timer expired
                    59    1D    DB     1F7B   3350          MFPR    #PR750$_CSRD,R9            ; dump receiver register 5.2
                  E07E CF       94     1F7E   3351          CLRB    B_FLAGS                    ; initialze flags
             54   FFF85EE0 8F   D0     1F82   3352 2$:      MOVL    #DELAY_TIME2,R4            ; load delay time
                    57    02    D0     1F89   3353          MOVL    #2,R7                      ; load byte count
               55   E08F  CF    9E     1F8C   3354          MOVAB   B_INIT_DRIVE,R5            ; load init flag
                0000203D'EF     16     1F91   3355          JSB     SEND_TU58                  ; send init
              28 E064 CF    00  E0     1F97   3356          BBS     #V_CLOCK_FLAG,B_FLAGS,10$ ; check for timeout error
             54   FFFB6C20 8F   D0     1F9D   3357          MOVL    #DELAY_TIME1,R4            ; load delay time
             55   00000000'FF   DE     1FA4   3358          MOVAL   @A_BUFFER_BEGIN,R5         ; load buffer address
               57    55    01   C1     1FAB   3359          ADDL3   #1,R5,R7                   ; load last expected address
                00002083'EF     16     1FAF   3360          JSB     RECEIVE_TU58               ; get continue from TU58
                    57    55    D1     1FB5   3361          CMPL    R5,R7                      ; check for correct number of bytes
                          05    13     1FB8   3362          BEQL    8$                         ; br = correct number of bytes
               E041 CF    02    88     1FBA   3363          BISB2   #M_RECEIVE_FLAG,B_FLAGS ; set receive flag to determine error
                    59    1D    DB     1FBF   3364 8$:      MFPR    #PR750$_CSRD,R9            ; dump receiver once more 5.2
                        0005    31     1FC2   3365          BRW     12$                        ; br to return
               E036 CF    10    88     1FC5   3366 10$:     BISB2   #M_SEND_FLAG,B_FLAGS       ; set send error flag
                                05     1FCA   3367 12$:     RSB                                ; return
                                       1FCB   3368
                                       1FCB   3369
                                       1FCB   3370 ; This subroutine will do an init sequence to the TU58. If the TU58
                                       1FCB   3371 ; does not respond or does not send a continue, an error will be
                                       1FCB   3372 ; reported in the main line code. This routine is called at the
                                       1FCB   3373 ; beginning of each TU58 test. When calling INIT_SEQUENCE R4 must
                                       1FCB   3374 ; contain the delay time for sending a break.
                                       1FCB   3375
                                       1FCB   3376 MINIT_SEQUENCE:
                  E031 CF       94     1FCB   3377          CLRB    B_FLAGS                    ; initialize timeout flag
             54   FFF85EE0 8F   D0     1FCF   3378          MOVL    #DELAY_TIME2,R4            ; load delay time
                    57    02    D0     1FD6   3379          MOVL    #2,R7                      ; load number of bytes to send
               55   E044  CF    9E     1FD9   3380          MOVAB   B_ZEROS,R5                 ; load address of data
                    1E    01    DA     1FDE   3381          MTPR    #1,#PR750$_CSTS            ; set break bit  5.2
                0000203D'EF     16     1FE1   3382          JSB     SEND_TU58                  ; send zeros
              4A E014 CF    00  E0     1FE7   3383 1$:      BBS     #V_CLOCK_FLAG,B_FLAGS,10$ ; exit if timer expired
                    59    1D    DB     1FED   3384          MFPR    #PR750$_CSRD,R9            ; dump receiver register 5.2
                  E00C CF       94     1FF0   3385          CLRB    B_FLAGS                    ; initialze flags
             54   FFF85EE0 8F   D0     1FF4   3386 2$:      MOVL    #DELAY_TIME2,R4            ; load delay time
                    57    02    D0     1FFB   3387          MOVL    #2,R7                      ; load byte count
               55   E01D  CF    9E     1FFE   3388          MOVAB   B_INIT_DRIVE,R5            ; load init flag
```

```
                0000203D'EF   16  2003   3389            JSB     SEND_TU58                  ; send init
            28 DFF2 CF    00  E0  2009   3390            BBS     #V_CLOCK_FLAG,B_FLAGS,10$ ; check for timeout error
           54   FFFB6C20 8F   D0  200F   3391            MOVL    #DELAY_TIME1,R4            ; load delay time
           55   00000000'FF   DE  2016   3392            MOVAL   @A_BUFFER_BEGIN,R5         ; load buffer address
                57   55   01  C1  201D   3393            ADDL3   #1,R5,R7                   ; load last expected address
                000020C8'EF   16  2021   3394            JSB     MRECEIVE_TU58              ; get continue from TU58
                     57   55  D1  2027   3395            CMPL    R5,R7                      ; check for correct number of bytes
                          05  13  202A   3396            BEQL    8$                         ; br = correct number of bytes
                DFCF CF   02  88  202C   3397            BISB2   #M_RECEIVE_FLAG,B_FLAGS ; set receive flag to determine error
                     59   1D  DB  2031   3398 8$:        MFPR    #PR750$_CSRD,R9            ; dump receiver once more   5.2
                        0005  31  2034   3399            BRW     12$                        ; br to return
                DFC4 CF   10  88  2037   3400 10$:       BISB2   #M_SEND_FLAG,B_FLAGS       ; set send error flag
                              05  203C   3401 12$:       RSB                                ; return
                                  203D   3402
                                  203D   3403
                                  203D   3404 ; This subroutine is used to transmit command and data packets to the
                                  203D   3405 ; TU58. When called the following general purpose registers are used:
                                  203D   3406 ;
                                  203D   3407 ;          R4 contains the initial time delay for a response from the TU58.
                                  203D   3408 ;          R5 contains the starting address of the data to be sent.
                                  203D   3409 ;          R6 is used to read the transmitter status and determine if ready is set.
                                  203D   3410 ;          R7 contains the number of bytes to be sent.
                                  203D   3411 ;          R8 is used as temporary data storage.
                                  203D   3412
                                  203D   3413 SEND_TU58:
                     DFBF CF  94  203D   3414            CLRB    B_FLAGS                    ; initialize timeout flag
                     19   54  DA  2041   3415 1$:        MTPR    R4,#PR750$_NICR            ; load clock with time 5.2
             00000000'8F  10  DA  2044   3416            MTPR    #M_TRANSFER,#PR$_ICCS      ; transfer time to ICR
                     19   01  DA  204B   3417            MTPR    #1,#PR750$_NICR            ; load next interval with long time 5.2
                     58   85  9A  204E   3418            MOVZBL  (R5)+,R8                   ; load byte to send into R8
                     1F   58  DA  2051   3419            MTPR    R8,#PR750$_CSTD            ; load byte to send 5.2
   00000000'8F   80000041 8F  DA  2054   3420            MTPR    #<M_ERROR+M_ENABLE+M_RUN>,#PR$_ICCS ; start timer
            1D DF9C CF    00  E0  205F   3421 2$:        BBS     #V_CLOCK_FLAG,B_FLAGS,3$ ; check for timeout
                     56   1E  DB  2065   3422            MFPR    #PR750$_CSTS,R6            ; read transmitter status 5.2
                  F3 56   07  E1  2068   3423            BBC     #V_READY,R6,2$             ; check for ready
   00000000'8F   80000080 8F  DA  206C   3424            MTPR    #<M_ERROR+M_INTERRUPT>,#PR$_ICCS ; ready, stop clock
           54   FFFB6C20 8F   D0  2077   3425            MOVL    #DELAY_TIME1,R4            ; load delay time
                          57  D7  207E   3426            DECL    R7                         ; count byte
                          BF  12  2080   3427            BNEQ    1$                         ; br = not finished
                              05  2082   3428 3$:        RSB                                ; return
                                  2083   3429
                                  2083   3430 ; This subroutine is used to receive data and end packets from the TU58.
                                  2083   3431 ; When called the following general purpose registers are used:
                                  2083   3432 ;
                                  2083   3433 ;          R4 contains the initial time delay for a response from the TU58.
                                  2083   3434 ;          R5 contains the starting buffer address of where the data is read to.
                                  2083   3435 ;          R6 is used to read the transmitter status and determine if ready is set.
                                  2083   3436 ;          R7 contains the expected end of receive buffer.
                                  2083   3437
                                  2083   3438 RECEIVE_TU58:
                     DF79 CF  94  2083   3439            CLRB    B_FLAGS                    ; initialize flags
                     19   54  DA  2087   3440 1$:        MTPR    R4,#PR750$_NICR            ; load clock with time 5.2
             00000000'8F  10  DA  208A   3441            MTPR    #M_TRANSFER,#PR$_ICCS      ; transfer time to ICR
                     19   01  DA  2091   3442            MTPR    #1,#PR750$_NICR            ; load next interval with long time 5.2
   00000000'8F   80000041 8F  DA  2094   3443            MTPR    #<M_ERROR+M_ENABLE+M_RUN>,#PR$_ICCS ; start timer
```

```
                  56   1C   DB  209F  3444 2$:     MFPR    #PR750$_CSRS,R6          ; read receiver status  5.2
          1F DF59 CF   00   E0  20A2  3445         BBS     #V_CLOCK_FLAG,B_FLAGS,3$ ; check for timeout
               F3 56   07   E1  20A8  3446         BBC     #V_DONE,R6,2$            ; check for done
    00000000'8F   80000080 8F   DA  20AC  3447         MTPR    #<M_ERROR+M_INTERRUPT>,#PR$_ICCS ; done, stop clock
                  65   1D   DB  20B7  3448         MFPR    #PR750$_CSRD,(R5)                  ; read byte  5.2
          54    FFF0BDC0 8F   D0  20BA  3449         MOVL    #DELAY_TIME4,R4               ; load delay time
          C0 55    DF4E CF   F2  20C1  3450         AOBLSS  BUF_LIM,R5,1$                 ; br = not finished, get another byte
                        05  20C7  3451 3$:     RSB                                   ; return
                            20C8  3452
                            20C8  3453
                            20C8  3454 ; This routine is used to receive data and end packets from tne new TU58.
                            20C8  3455 ; The new TU58 have MRSP instead of RSP.
                            20C8  3456 ;
                            20C8  3457 MRECEIVE_TU58:
                DF34 CF   94  20C8  3458         CLRB    B_FLAGS                       ;initialize flags
                  19   54   DA  20CC  3459 1$:     MTPR    R4,#PR750$_NICR               ;load clock with time 5.2
         00000000'8F   10   DA  20CF  3460         MTPR    #M_TRANSFER,#PR$_ICCS   ;transfer time ti ICR
                  19   01   DA  20D6  3461         MTPR    #1,#PR750$_NICR               ;load next interval with long time 5.2
    00000000'8F   80000041 8F   DA  20D9  3462         MTPR    #<M_ERROR+M_ENABLE+M_RUN>,#PR$_ICCS  ;strat timer
                  56   1C   DB  20E4  3463 2$:     MFPR    #PR750$_CSRS,R6               ;read receiver status  5.2
          46 DF14 CF   00   E0  20E7  3464         BBS     #V_CLOCK_FLAG,B_FLAGS,3$  ;check for timeout
               F3 56   07   E1  20ED  3465         BBC     #V_DONE,R6,2$                 ;check for done
    00000000'8F   80000080 8F   DA  20F1  3466         MTPR    #<M_ERROR+M_INTERRUPT>,#PR$_ICCS  ;done, stop clock
                  65   1D   DB  20FC  3467         MFPR    #PR750$_CSRD,(R5)                    ;read byte 5.2
             DF13 CF   55   D0  20FF  3468         MOVL    R5,TEMP5                      ;store R5
             DF12 CF   57   D0  2104  3469         MOVL    R7,TEMP7                      ;store R7
          54    FFFB6C20 8F   D0  2109  3470         MOVL    #DELAY_TIME1,R4               ;load delay time
                  57   01   D0  2110  3471         MOVL    #1,R7                         ;number of byte to be send
             55    DF0C CF   9E  2113  3472         MOVAB   B_CONTINUE,R5                 ;point to data address
                FF21 CF   16  2118  3473         JSB     SEND_TU58                     ;send data to TU58
             57    DEFB CF   D0  211C  3474         MOVL    TEMP7,R7                      ;restore R7
             55    DEF2 CF   D0  2121  3475         MOVL    TEMP5,R5                      ;restore R5
          54    FFF0BDC0 8F   D0  2126  3476         MOVL    #DELAY_TIME4,R4               ;load delay time
          99 55    DEE2 CF   F2  212D  3477         AOBLSS  BUF_LIM,R5,1$                 ;br = not finished, get another byte
                        05  2133  3478 3$:     RSB                                   ;return
                            2134  3479
                            2134  3480
                            2134  3481 ; This routine is used to clear a specified buffer area. The following
                            2134  3482 ; general purpose registers are used:
                            2134  3483 ;
                            2134  3484 ;       R8 contains the starting address of the buffer to be cleared.
                            2134  3485 ;       R9 contains the number of bytes to be cleared.
                            2134  3486
                            2134  3487 CLEAR_BUFFER:
                       88   94  2134  3488         CLRB    (R8)+                         ; clear a byte
                       59   D7  2136  3489         DECL    R9                            ; count a byte
                       FA   12  2138  3490         BNEQ    CLEAR_BUFFER                  ; br = not finished
                            05  213A  3491         RSB                                   ; return
                            213B  3492
                            213B  3493 ; This subroutine is used to compare data and end packets read from the
                            213B  3494 ; TU58 with expected data to determine if an error exists. The data is
                            213B  3495 ; compared on a byte to byte level and sets a flag to signal the caller
                            213B  3496 ; that an error was encountered. The following general purpose registers
                            213B  3497 ; are used:
                            213B  3498 ;
```

```
                                     213B  3499 ;       R0 is used as the error flag. R0 = 0, no error. R0 = 1, error.
                                     213B  3500 ;       R10 contains the starting address of the expected data.
                                     213B  3501 ;       R11 contains the starting address of the actual data.
                                     213B  3502 ;       R12 contains the number of bytes to check.
                                     213B  3503
                                     213B  3504 CHECK_PACKET:
                          50    7C   213B  3505         CLRQ    R0                      ; initialize error flag and offset
                6B41    6A41    91   213D  3506 1$:     CMPB    (R10)[R1],(R11)[R1]     ; check a byte
                          06    12   2142  3507         BNEQ    2$                      ; br = error
                  F5 51   5C    F2   2144  3508         AOBLSS  R12,R1,1$               ; increment offset, br = not done
                          02    11   2148  3509         BRB     3$                      ; br = done
                          50    D6   214A  3510 2$:     INCL    R0                      ; flag error
                                05   214C  3511 3$:     RSB                             ; return
                                     214D  3512
                                     214D  3513
                                     214D  3514 ; This interrupt service routine is used to set CLOCK_FLAG in the event
                                     214D  3515 ; of a timeout. Is also used to determine when a wait loop time has been
                                     214D  3516 ; satisfied.
                                     214D  3517
                                     214D  3518         .ALIGN  LONG
                                     2150  3519 TIMER_SERVICE:
                     00'  00    DA   2150  3520         MTPR    #0,S^#PR$_ICCS          ; Turn off the timer
     00000000'8F  80000080 8F   DA   2153  3521         MTPR    #<M_ERROR+M_INTERRUPT>,#PR$_ICCS ; stop clock
             DE9D CF      01    88   215E  3522         BISB2   #M_CLOCK_FLAG,B_FLAGS   ; set timeout error flag
                                02   2163  3523         REI                             ; return from interrupt
                                     2164  3524
                                     2164  3525
                                     2164  3526
                                     2164  3527 ; This interrupt-service routine reads a byte from the TU58 into the
                                     2164  3528 ; location pointed to by R3
                                     2164  3529
                                     2164  3530         .ALIGN LONG
                                     2164  3531 READ_SERVICE:
                     63   1D    DB   2164  3532         MFPR    #PR750$_CSRD,(R3)               ; read byte  5.2
                          53    D6   2167  3533         INCL    R3                      ; increment pointer
                                02   2169  3534         REI                             ; return
                                     216A  3535
                                     216A  3536 ; This interrupt-service routine reads a byte from the TU58 into the
                                     216A  3537 ; location pointed to by R3 and send a CONTINUE to TU58
                                     216A  3538
                                     216A  3539         .ALIGN LONG
                                     216C  3540 MREAD_SERVICE:
                     56   1C    DB   216C  3541 2$:     MFPR    #PR750$_CSRS,R6         ; read receive status 5.2
                  F9 56   07    E1   216F  3542         BBC     #V_DONE,R6,2$           ; check for done
                     63   1D    DB   2173  3543         MFPR    #PR750$_CSRD,(R3)               ; read byte 5.2
                          53    D6   2176  3544         INCL    R3                      ; increment pointer
             DE9A CF      55    D0   2178  3545         MOVL    R5,TEMP5                ; store R5
             DE99 CF      57    D0   217D  3546         MOVL    R7,TEMP7                ; store R7
             55    DE9D CF      9E   2182  3547         MOVAB   B_CONTINUE,R5           ; point to data address
                     58   65    9A   2187  3548         MOVZBL  (R5),R8                 ; load byte to sendinot R8
                     1F   58    DA   218A  3549         MTPR    R8,#PR750$_CSTD         ; load byte to send 5.2
                     57   1E    DB   218D  3550 3$:     MFPR    #PR750$_CSTS,R7         ; read transmitter status 5.2
                  F9 57   07    E1   2190  3551         BBC     #V_READY,R7,3$          ; check for ready
             55    DE7F CF      D0   2194  3552         MOVL    TEMP5,R5                ; restore R5
             57    DE7E CF      D0   2199  3553         MOVL    TEMP7,R7                ; restore R7
```

```
                                  02  219E  3554          REI                                  ; return
                                      219F  3555
                                      219F  3556
                                      219F  3557 ; This interrupt-service routine sets the read interrupt flag and disables
                                      219F  3558 ; the read interrupt.
                                      219F  3559
                                      219F  3560          .ALIGN LONG
                                      21A0  3561 READ_SERVICE2:
                  DE5D CF     04  88  21A0  3562          BISB2   #M_RD_INT_FLAG,B_FLAGS3 ; set read interrupt flag
                       1^     00  DA  21A5  3563          MTPR    #0,#PR750$_CSRS         ; disable read interrupt 5.2
                                  02  21A8  3564          REI                             ; return
                                      21A9  3565
                                      21A9  3566
                                      21A9  3567 ; This interrupt service routine set the read interruptflag and send
                                      21A9  3568 ; a continue to TU58
                                      21A9  3569
                                      21A9  3570          .ALIGN  LONG
                                      21AC  3571 MREAD_SERVICE2:
                  DE51 CF     04  88  21AC  3572          BISB2   #M_RD_INT_FLAG,B_FLAGS3 ; set read interrupt flag
                       56     1C  DB  21B1  3573 2$:      MFPR    #PR750$_CSRS,R6         ; read receive status          5.2
                    F9 56     07  E1  21B4  3574          BBC     #V_DONE,R6,2$           ; check for done
                       63     1D  DB  21B8  3575          MFPR    #PR750$_CSRD,(R3)               ; read byte            5.2
                              53  D6  21BB  3576          INCL    R3                      ; increment pointer
                  55   DE62 CF    9E  21BD  3577          MOVAB   B_CONTINUE,R5           ; point to data address
                       58     65  9A  21C2  3578          MOVZBL  (R5),R8                 ; load byte to sendinot R8
                       1F     58  DA  21C5  3579          MTPR    R8,#PR750$_CSTD         ; load byte to send            5.2
                       57     1E  DB  21C8  3580 3$:      MFPR    #PR750$_CSTS,R7         ; read transmitter status      5.2
                    F9 57     07  E1  21CB  3581          BBC     #V_READY,R7,3$          ; check for ready
                       FD86 CF    16  21CF  3582          JSB     INIT_SEQUENCE           ; initialize driver
                                  02  21D3  3583          REI                             ; return
                                      21D4  3584
                                      21D4  3585
                                      21D4  3586
                                      21D4  3587 ; This interrupt-service routine sends a byte to the TU58 from the
                                      21D4  3588 ; location pointed to by R5 and increments R5.
                                      21D4  3589
                                      21D4  3590          .ALIGN LONG
                                      21D4  3591 WRITE_SERVICE:
                       1F     65  DA  21D4  3592          MTPR    (R5),#PR750$_CSTD               ; send byte            5.2
                              55  D6  21D7  3593          INCL    R5                      ; increment pointer
                       57     55  D1  21D9  3594          CMPL    R5,R7                   ; finished sending ?
                              08  19  21DC  3595          BLSS    20$                     ; br = not finished
                  DE1F CF     02  88  21DE  3596          BISB2   #M_SDONE_FLAG,B_FLAGS3  ; set send done flag
                       1E     00  DA  21E3  3597          MTPR    #0,#PR750$_CSTS         ; disable send interrupt       5.2
                                  02  21E6  3598 20$:     REI                             ; return
                                      21E7  3599
                                      21E7  3600
                                      21E7  3601 ; This routine prints the expected data, the actual data, and an XOR
                                      21E7  3602 ; for each erroneous byte for the packets described as follows:
                                      21E7  3603
                                      21E7  3604 ;  R10 contains the address of the first byte of expected data
                                      21E7  3605 ;  R11 contains the address of the first byte of actual data
                                      21E7  3606 ;  R12 contains the number of bytes in the packet
                                      21E7  3607
                                      21E7  3608 DUMP_PKT:
```

```
                      59    D4  21E7  3609          CLRL    R9                      ; clear offset
                                21E9  3610          $DS_PRINTX_S    FORMAT=T_DUMP_HDR ; print header
                FAD1 CF     9F  21E9                        PUSHAB  T_DUMP_HDR
          000100E8 9F   01  FB  21ED                        CALLS   #$$N, @#DS$PRINTX
                 6B   6A    91  21F4  3611 1$:      CMPB    (R10),(R11)                     ; correct byte ?
                      17    13  21F7  3612          BEQL    2$                              ; br = correct (no XOR required)
            58   6B   6A    CD  21F9  3613          XORL3   (R10),(R11),R8                  ; load XOR
                                21FD  3614          $DS_PRINTX_S    FORMAT=T_DUMP_XOR,P1=(R10),P2=(R11),P3=R8 ; dump bytes
                      58    DD  21FD                        PUSHL   R8
                      6B    DD  21FF                        PUSHL   (R11)
                      6A    DD  2201                        PUSHL   (R10)
                FAD3 CF     9F  2203                        PUSHAB  T_DUMP_XOR
          000100E8 9F   04  FB  2207                        CALLS   #$$N, @#DS$PRINTX
                      0F    11  220E  3615          BRB     3$                              ; do next byte
                                2210  3616 2$:      $DS_PRINTX_S    FORMAT=T_DUMP_NOXOR,P1=(R10),P2=(R11) ; dump bytes
                      6B    DD  2210                        PUSHL   (R11)
                      6A    DD  2212                        PUSHL   (R10)
                FADA CF     9F  2214                        PUSHAB  T_DUMP_NOXOR
          000100E8 9F   03  FB  2218                        CALLS   #$$N, @#DS$PRINTX
                 8B   8A    91  221F  3617 3$:      CMPB    (R10)+,(R11)+                   ; increment pointers
              CE 59   5C    F2  2222  3618          AOBLSS  R12,R9,1$                       ; increment count and br = not done
                            05  2226  3619          RSB                                     ; return
                                2227  3620
                                2227  3621
                                2227  3622 ; This subroutine dumps 14 bytes beginning at the location contained in R9.
                                2227  3623
                                2227  3624 DUMP_14:
                      58    D4  2227  3625          CLRL    R8                              ; clear dump count
                                2229  3626 1$:      $DS_PRINTX_S    FORMAT=T_DUMP_14,P1=(R9) ; print byte
                      69    DD  2229                        PUSHL   (R9)
                FC13 CF     9F  222B                        PUSHAB  T_DUMP_14
          000100E8 9F   02  FB  222F                        CALLS   #$$N, @#DS$PRINTX
                      59    D6  2236  3627          INCL    R9                              ; increment pointer
              ED 58   0E    F2  2238  3628          AOBLSS  #14,R8,1$                       ; increment count and br = not done
                            05  223C  3629          RSB                                     ; return
                                223D  3630
                                223D  3631
                                223D  3632 ; This subroutine checks whether the 14 bytes preceding the location
                                223D  3633 ; contained in R7 contain and end flag and the correct byte count to be
                                223D  3634 ; interpreted as an end packet.  R0 is set if successful; else clear.
                                223D  3635
                                223D  3636 FIND_END:
                      50    D4  223D  3637          CLRL    R0                              ; clear success code
            59   55   0E    C3  223F  3638          SUBL3   #14,R5,R9                       ; load address of word to check
            E418 CF   69    B1  2243  3639          CMPW    (R9),MB_END_PACKET_1            ; is this an end packet ?
                      02    12  2248  3640          BNEQ    10$                             ; br = not end packet
                      50    D6  224A  3641          INCL    R0                              ; set success code
                            05  224C  3642 10$:     RSB                                     ; return
                                224D  3643
                                224D  3644
                                224D  3645
                                224D  3646          $DS PAGE
```

```
                                     224D  3648
                                     224D                .SBTTL  TEST 9:  TU58 EXERCISER
                                 00000000                        .PSECT  TEST_009, PAGE, NOWRT
                                     0014
                                     0014  3650 ;++
                                     0014  3651 ; Test description:
                                     0014  3652 ;
                                     0014  3653 ;       This test exercises the TU58 on the VAX-11/750 by performing
                                     0014  3654 ;       initialization, self diagnostic and no operation.
                                     0014  3655 ;       All information received from the TU58 is checked for errors.
                                     0014  3656 ;       All errors detected are reported.
                                     0014  3657 ;
                                     0014  3658 ;
                                     0014  3659 ; Assumptions:
                                     0014  3660 ;
                                     0014  3661 ;       This subtest does not write or read the TU58. All operations performed
                                     0014  3662 ;       deal with the controller only. (NOTE: Test 2 requires a scratch tape).
                                     0014  3663 ;
                                     0014  3664 ;
                                     0014  3665 ;.page
                                     0014  3666 ;--
                                     0014  3667
                                     0014  3668         $DS_BGNTEST     <DEFAULT,TU58,ALL>
                                     0014       DATA_009:
                           00000000  0014                        .LONG   0               ; TEST ARGUMENT TABLE TERMINATOR
                                     0018       TEST_009::
                               0000  0018                        .WORD   ^M<>            ; ENTRY MASK
                                     001A  3669
                                     001A  3670
                                     001A  3671         ; Set Up
                                     001A  3672
                                     001A  3673
                                     001A  3674         $DS_SETVEC_S    VECTOR=#SCB$L_TIMER,SRVADR=TIMER_SERVICE
                            00   DD  001A                        PUSHL   #0
                   00002150'EF   DF  001C                        PUSHAL  TIMER_SERVICE
                   000000C0 8F   DD  0022                        PUSHL   #SCB$L_TIMER
              00010160 9F   03   FB  0028                        CALLS   #3, @#DS$SETVEC
              58   00000000'FF   9E  002F  3675 1$:     MOVAB   @A_BUFFER_BEGIN,R8      ; load buffer starting address
00000013'EF   58   00000092 8F   C1  0036  3676         ADDL3   #146,R8,BUF_LIM         ; construct upper buffer limit
              59   00000400 8F   D0  0042  3677         MOVL    #1024,R9                ; do both pages
                   00002134'EF   16  0049  3678         JSB     CLEAR_BUFFER            ; go clear buffers
                   00000000'EF   94  004F  3679         CLRB    B_FLAGS                 ; init flags
                   00000001'EF   94  0055  3680         CLRB    B_FLAGS2                ; init flags
                                     005B  3681
                   00001F59'EF   16  005B  3682 2$:     JSB     INIT_SEQUENCE           ; initialize drive
              5A   00000023'EF   9E  0061  3683         MOVAB   B_CONTINUE,R10          ; load expected data address
              5B   00000000'FF   9E  0068  3684         MOVAB   @A_BUFFER_BEGIN,R11     ; load actual data address
           2C 00000000'EF   04   E1  006F  3685         BBC     #V_SEND_FLAG,B_FLAGS,5$ ; br = no send error
           DC 00000001'EF   01   E3  0077  3686         BBCS    #V_REPEAT_SEND,B_FLAGS2,2$ ; br = first try and set flag
                                     007F  3687         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_INIT1_ERR,PRLINK=NO_READY
                   00001BEB'EF   DF  007F                        PUSHAL  NO_READY
                   0000183B'EF   DF  0085                        PUSHAL  T_INIT1_ERR
                            00   DD  008B                        PUSHL   #0
                            01   DD  008D                        PUSHL   #$ER
                                     008F       ;;; TEST 9, SUBTEST 0, ERROR 1
              000100D0 9F   04    FB  008F                       CALLS   #$$M, @#DS$ERRHARD
                                     0096  3688         $DS_BNCOMPLETE  3$
                         03 50    E9  0096                       BLBC    R0, 3$
                                     0099  3689         $DS_ABORT       TEST
                            50    D4  0099                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                  04  009B                       RET                     ; TERMINATE TEST
                                     009C  3690 3$:     $DS_ABORT
              00010020 9F   6C    FA  009C                       CALLG   (AP), @#DS$ABORT
                                     00A3  3691
           32 00000000'EF   01    E1  00A3  3692 5$:     BBC     #V_RECEIVE_FLAG,B_FLAGS,9$ ; br = no receive error
           A8 00000001'EF   01    E3  00AB  3693          BBCS    #V_REPEAT_SEND,B_FLAGS2,2$ ; br = first try and set flag
                                     00B3  3694 7$:     $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_INIT1_ERR,PRLINK=REC_ERR,-
                                     00B3  3695                         P1=#T_REC_CONT
                   00001D47'8F    DD  00B3                       PUSHL   #T_REC_CONT
                   00001EB4'EF    DF  00B9                       PUSHAL  REC_ERR
                   0000183B'EF    DF  00BF                       PUSHAL  T_INIT1_ERR
                            00    DD  00C5                       PUSHL   #0
                            02    DD  00C7                       PUSHL   #$ER
                                     00C9       ;;; TEST 9, SUBTEST 0, ERROR 2
              000100D0 9F   05    FB  00C9                       CALLS   #$$M, @#DS$ERRHARD
                                     00D0  3696         $DS_BNCOMPLETE  8$
                         03 50    E9  00D0                       BLBC    R0, 8$
                                     00D3  3697         $DS_ABORT       TEST
                            50    D4  00D3                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                  04  00D5                       RET                     ; TERMINATE TEST
                                     00D6  3698 8$:     $DS_ABORT
              00010020 9F   6C    FA  00D6                       CALLG   (AP), @#DS$ABORT
                                     00DD  3699
                       5C   01    D0  00DD  3700 9$:     MOVL    #1,R12                  ; load number of bytes to check
                                     00E0  3701         $DS_BREAK
              00010058 9F   6E    FA  00E0                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF    16  00E7  3702         JSB     CHECK_PACKET            ; go check continue
                                     00ED  3703         $DS_BNCOMPLETE  12$             ; br = continue received
                         39 50    E9  00ED                       BLBC    R0, 12$
           0A 00000001'EF   01    E0  00F0  3704          BBS     #V_REPEAT_SEND,B_FLAGS2,10$ ; br = first try
              00000001'EF   02    88  00F8  3705          BISB2   #M_REPEAT_SEND,B_FLAGS2 ; set repeat flag
                          FF59    31  00FF  3706          BRW     2$                      ; try again
                       52   50    D0  0102  3707 10$:    MOVL    R0,R2                   ; save code for error report
                                     0105  3708         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_INIT1_ERR,PRLINK=DATA_ERR_CONT
                   00001E4D'EF    DF  0105                       PUSHAL  DATA_ERR_CONT
                   0000183B'EF    DF  010B                       PUSHAL  T_INIT1_ERR
                            00    DD  0111                       PUSHL   #0
                            03    DD  0113                       PUSHL   #$ER
                                     0115       ;;; TEST 9, SUBTEST 0, ERROR 3
              000100D0 9F   04    FB  0115                       CALLS   #$$M, @#DS$ERRHARD
                                     011C  3709         $DS_BNCOMPLETE  11$
                         03 50    E9  011C                       BLBC    R0, 11$
                                     011F  3710         $DS_ABORT       TEST
                            50    D4  011F                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                  04  0121                       RET                     ; TERMINATE TEST
                                     0122  3711 11$:    $DS_ABORT
              00010020 9F   6C    FA  0122                       CALLG   (AP), @#DS$ABORT
                                     0129  3712 12$:
                                     0129  3713         $DS_PAGE
```

```
0129              .SBTTL  SUBTEST 9.1:   INITIALIZE
0129        ;+
0129        ; SUBTEST DESCRIPTION:
0129        ;
0129        ;       This subtest sends the TU58 a command packet to perform an initialization
0129        ;       operation which puts the controller into a known ready state. While
0129        ;       sending each byte of the packet, a timer is set which will time out if
0129        ;       the ready bit in the transmitter processor register does not set
0129        ;       within the alloted time. The same is done for the done bit in the
0129        ;       receiver processor register while reading a byte from the TU58. When
0129        ;       this is completed successfully, the returned end packet is compared to
0129        ;       an expected end packet. If a time out occurs or bad data is
0129        ;       encountered an error will be reported.
0129        ;
0129        ;
0129        ; SUBTEST STEPS:
0129        ;
0129        ;       SUBTEST INITIALIZE
0129        ;       :  Clear buffer
0129        ;       :  Point to initialize command packet
0129        ;       :  REPEAT
0129        ;       :  :  Send command packet byte at a time
0129        ;       :  :  Set timer
0129        ;       :  :  IF timer expired AND no response
0129        ;       :  :  :  THEN
0129        ;       :  :  :  Report no response error
0129        ;       :  :  :  Exit test
0129        ;       :  :  ENDIF
0129        ;       :  : UNTIL all bytes sent
0129        ;       :  ENDREPEAT
0129        ;       :  REPEAT
0129        ;       :  :  Set timer
0129        ;       :  :  IF timer expired AND done set
0129        ;       :  :  :  THEN
0129        ;       :  :  :  :  Read byte
0129        ;       :  :  ENDIF
0129        ;       :  : UNTIL timer expired AND done clear
0129        ;       :  ENDREPEAT
0129        ;       :  IF expected byte count NEQ actual byte count
0129        ;       :  :  THEN
0129        ;       :  :  :  Report error
0129        ;       :  :  :  Exit test
0129        ;       :  ENDIF
0129        ;       :  Point to expected and actual end packet
0129        ;       :  REPEAT
0129        ;       :  :  IF data expected NEQ data received
0129        ;       :  :  :  THEN
0129        ;       :  :  :  Report error
0129        ;       :  :  :  Exit test
0129        ;       :  :  ENDIF
0129        ;       :  : UNTIL all data checked
0129        ;       :  ENDREPEAT
0129        ;       ENDSUBTEST INITIALIZE
0129        ;
0129        ;
```

```
                              0129  3770 ;
                              0129  3771 ;
                              0129  3772 ;.page
                              0129  3773 ;-
                              0129  3774
                              0129  3775            $DS_BGNSUB
                              0129       T9_S1::
    00010030 9F   00000000'EF   FA  0129                      CALLG   $S$, a#DS$BGNSUB
                              0134  3776
               00000000'EF   94  0134  3777            CLRB    B_FLAGS                   ; initialize flags
          58   00000000'FF   9E  013A  3778            MOVAB   aA_BUFFER_BEGIN,R8        ; load buffer starting address
          59   00000400 8F   D0  0141  3779            MOVL    #1024,R9                  ; do both pages
               00002134'EF   16  0148  3780            JSB     CLEAR_BUFFER              ; go clear buffers
          54   FFFB6C20 8F   D0  014E  3781            MOVL    #DELAY_TIME1,R4           ; load send packet delay time
          55   00000024'EF   9E  0155  3782            MOVAB   B_INIT_CMD,R5             ; point to init command packet
                     57  0E  D0  015C  3783            MOVL    #14,R7                    ; load transfer byte count
               0000203D'EF   16  015F  3784            JSB     SEND_TU58                 ; send command packet
       24 00000000'EF    00  E1  0165  3785            BBC     #V_CLOCK_FLAG,B_FLAGS,2$  ; br = no timeout error
                              016D  3786            $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_INIT2_ERR,PRLINK=NO_READY
               00001BEB'EF   DF  016D                      PUSHAL  NO_READY
               00001865'EF   DF  0173                      PUSHAL  T_INIT2_ERR
                         00  DD  0179                      PUSHL   #0
                         01  DD  017B                      PUSHL   #$ER
                              017D       ;;; TEST 9, SUBTEST 1, ERROR 1
          000100D0 9F    04  FB  017D                      CALLS   #$$M, a#DS$ERRHARD
                              0184  3787            $DS_BNCOMPLETE  1$
                      03 50  E9  0184                      BLBC    R0, 1$
                              0187  3788            $DS_ABORT       TEST
                         50  D4  0187                      CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                         04  0189                          RET                         ; TERMINATE TEST
                              018A  3789 1$:        $DS_ABORT
          00010020 9F    6C  FA  018A                      CALLG   (AP), a#DS$ABORT
                              0191  3790
          54   FFFB6C20 8F   D0  0191  3791 2$:     MOVL    #DELAY_TIME1,R4           ; load receive packet delay time
          55   00000000'FF   9E  0198  3792            MOVAB   aA_BUFFER_BEGIN,R5        ; load buffer address to store data
              57   55    0E  C1  019F  3793            ADDL3   #14,R5,R7                 ; load last expected address of data
               00002083'EF   16  01A3  3794            JSB     RECEIVE_TU58              ; go get data from TU58
          5A   000005D4'EF   9E  01A9  3795            MOVAB   B_END_PACKET_1,R10        ; load address of expected end packet
          5B   00000000'FF   9E  01B0  3796            MOVAB   aA_BUFFER_BEGIN,R11       ; load address of actual end packet
                     57  55  D1  01B7  3797            CMPL    R5,R7                     ; check for correct number of bytes
                         31  13  01BA  3798            BEQL    6$                        ; br = correct number of bytes
          00000000'EF    08  88  01BC  3799            BISB2   #M_END_FLAG,B_FLAGS       ; set end error flag
                              01C3  3800 4$:        $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_INIT2_ERR,PRLINK=REC_ERR,-
                              01C3  3801                             P1=#T_REC_END
               00001D2E'8F   DD  01C3                      PUSHL   #T_REC_END
               00001EB4'EF   DF  01C9                      PUSHAL  REC_ERR
               00001865'EF   DF  01CF                      PUSHAL  T_INIT2_ERR
                         00  DD  01D5                      PUSHL   #0
                         02  DD  01D7                      PUSHL   #$ER
                              01D9       ;;; TEST 9, SUBTEST 1, ERROR 2
          000100D0 9F    05  FB  01D9                      CALLS   #$$M, a#DS$ERRHARD
                              01E0  3802            $DS_BNCOMPLETE  5$
                      03 50  E9  01E0                      BLBC    R0, 5$
                              01E3  3803            $DS_ABORT       TEST
                         50  D4  01E3                      CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
```

```
                              04  01E5                  RET                               ; TERMINATE TEST
                                  01E6  3804 5$:        $DS_ABORT
          00010020 9F    6C   FA  01E6                  CALLG   (AP), a#DS$ABORT
                                  01ED  3805
                   5C    0E   D0  01ED  3806 6$:        MOVL    #14,R12                   ; load number of bytes to check
                                  01F0  3807            $DS_BREAK
          00010058 9F    6E   FA  01F0                  CALLG   (SP), a#DS$BREAK
               0000213B'EF   16   01F7  3808            JSB     CHECK_PACKET              ; go check end packet
                                  01FD  3809            $DS_BNCOMPLETE  7$                ; br = no error
                      21 50   E9  01FD                  BLBC    R0, 7$
                                  0200  3810            $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_INIT2_ERR,PRLINK=DATA_ERR_END
               00001E9D'EF   DF   0200                  PUSHAL  DATA_ERR_END
               00001865'EF   DF   0206                  PUSHAL  T_INIT2_ERR
                         00   DD  020C                  PUSHL   #0
                         03   DD  020E                  PUSHL   #$ER
                                  0210       ;;; TEST 9, SUBTEST 1, ERROR 3
          000100D0 9F    04   FB  0210                  CALLS   #$$M, a#DS$ERRHARD
                                  0217  3811            $DS_BCOMPLETE   7$                ; continue if only data error
                      07 50   E8  0217                  BLBS    R0, 7$
                                  021A  3812            $DS_ABORT
          00010020 9F    6C   FA  021A                  CALLG   (AP), a#DS$ABORT
                                  0221  3813 7$:
                                  0221  3814
                                  0221  3815            $DS_ENDSUB
                                  0221       T9_S1_X:
 00010038 9F   00000000'EF    FA  0221                  CALLG   $S$, a#DS$ENDSUB
                                  022C  3816
                                  022C  3817            $DS_PAGE
```

```
022C              .SBTTL  SUBTEST 9.2:   SELF DIAGNOSTIC
022C  3820 ;+
022C  3821 ; SUBTEST DESCRIPTION:
022C  3822 ;
022C  3823 ;           This subtest sends the TU58 a command packet to run its internal
022C  3824 ;           diagnostic. While sending each byte of the packet,a timer is set which
022C  3825 ;           will time out if the ready bit in the transmitter processor register
022C  3826 ;           does not set within the alloted time. The same is done for the done
022C  3827 ;           bit in the receiver processor register while reading a byte from the
022C  3828 ;           TU58. When this is completed successfully, the returned end packet is
022C  3829 ;           compared to an expected end packet. If a time out occurs or bad data
022C  3830 ;           is encountered an error will be reported.
022C  3831 ;
022C  3832 ;
022C  3833 ; SUBTEST STEPS:
022C  3834 ;
022C  3835 ;           SUBTEST SELF DIAGNOSTIC
022C  3836 ;           :  Clear buffer
022C  3837 ;           :  Point to self diagnostic command packet
022C  3838 ;           :  REPEAT
022C  3839 ;           :  :  Send command packet byte at a time
022C  3840 ;           :  :  Set timer
022C  3841 ;           :  :  IF timer expired AND no response
022C  3842 ;           :  :  :  THEN
022C  3843 ;           :  :  :  Report no response error
022C  3844 ;           :  :  :  Exit test
022C  3845 ;           :  :  ENDIF
022C  3846 ;           :  : UNTIL all bytes sent
022C  3847 ;           :  ENDREPEAT
022C  3848 ;           :  REPEAT
022C  3849 ;           :  :  Set timer
022C  3850 ;           :  :  IF timer expired AND done set
022C  3851 ;           :  :  :  THEN
022C  3852 ;           :  :  :  :  Read byte
022C  3853 ;           :  :  ENDIF
022C  3854 ;           :  : UNTIL timer expired AND done clear
022C  3855 ;           :  ENDREPEAT
022C  3856 ;           :  IF expected byte count NEQ actual byte count
022C  3857 ;           :  :  THEN
022C  3858 ;           :  :  :  Report error
022C  3859 ;           :  :  :  Exit test
022C  3860 ;           :  ENDIF
022C  3861 ;           :  Point to expected and actual data
022C  3862 ;           :  REPEAT
022C  3863 ;           :  :  IF expected data NEQ received data
022C  3864 ;           :  :  :  THEN
022C  3865 ;           :  :  :  Report data error
022C  3866 ;           :  :  :  Exit test
022C  3867 ;           :  :  ENDIF
022C  3868 ;           :  : UNTIL all data checked
022C  3869 ;           :  ENDREPEAT
022C  3870 ;           ENDSUBTEST SELF DIAGNOSTIC
022C  3871 ;
022C  3872 ;
022C  3873 ;
```

```
                                      022C  3874 ;.page
                                      022C  3875 ;-
                                      022C  3876          $DS_BGNSUB
                                      022C       T9_S2::
     00010030 9F    0000000C'EF   FA  022C                         CALLG   $S$, a#DS$BGNSUB
                                      0237  3877
                                      0237  3878
                    00000000'EF   94  0237  3879          CLRB     B_FLAGS                      ; initialize flags
               58   00000000'FF   9E  023D  3880          MOVAB    aA_BUFFER_BEGIN,R8           ; load buffer starting address
               59   00000400 8F   D0  0244  3881          MOVL     #1024,R9                     ; do both pages
                    00002134'EF   16  024B  3882          JSB      CLEAR_BUFFER                 ; go clear buffers
               54   FFFB6C20 8F   D0  0251  3883          MOVL     #DELAY_TIME1,R4              ; load send packet delay time
               55   00000040'EF   9E  0258  3884          MOVAB    B_DIAGNOSE_CMD,R5            ; point to diagnose command packet
                         57  0E   D0  025F  3885          MOVL     #14,R7                       ; load transfer byte count
                    0000203D'EF   16  0262  3886          JSB      SEND_TU58                    ; send command packet
            24 00000000'EF   00   E1  0268  3887          BBC      #V_CLOCK_FLAG,B_FLAGS,2$ ; br = no timeout error
                                      0270  3888          $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_SLF_DIAG_ERR,PRLINK=NO_READY
                    00001BEB'EF   DF  0270                         PUSHAL  NO_READY
                    00001888'EF   DF  0276                         PUSHAL  T_SLF_DIAG_ERR
                             00   DD  027C                         PUSHL   #0
                             01   DD  027E                         PUSHL   #$ER
                                      0280       ;;; TEST 9, SUBTEST 2, ERROR 1
          000100D0 9F        04   FB  0280                         CALLS   #$$M, a#DS$ERRHARD
                                      0287  3889          $DS_BNCOMPLETE  1$
                          03 50   E9  0287                         BLBC    R0, 1$
                                      028A  3890          $DS_ABORT       TEST
                             50   D4  028A                         CLRL    R0                   ; SEND A WARNING TO THE SUPERVISOR
                             04   028C                             RET                          ; TERMINATE TEST
                                      028D  3891 1$:      $DS_ABORT
          00010020 9F        6C   FA  028D                         CALLG   (AP), a#DS$ABORT
                                      0294  3892
          54        FFF3CB00 8F   D0  0294  3893 2$:      MOVL     #DELAY_TIME3,R4              ; load receive packet delay time
          55        00000000'FF   9E  029B  3894          MOVAB    aA_BUFFER_BEGIN,R5           ; load buffer address to store data
               57   55       0E   C1  02A2  3895          ADDL3    #14,R5,R7                    ; load last expected address of data
                    00002083'EF   16  02A6  3896          JSB      RECEIVE_TU58                 ; go get data from TU58
          5A        000005D4'EF   9E  02AC  3897          MOVAB    B_END_PACKET_1,R10           ; load address of expected end packet
          5B        00000000'FF   9E  02B3  3898          MOVAB    aA_BUFFER_BEGIN,R11          ; load address of actual end packet
                    57       55   D1  02BA  3899          CMPL     R5,R7                        ; check for expected number of bytes
                             31   13  02BD  3900          BEQL     6$                           ; br = correct number of bytes
          00000000'EF        08   88  02BF  3901          BISB2    #M_END_FLAG,B_FLAGS          ; set end error flag
                                      02C6  3902 4$:      $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_SLF_DIAG_ERR,PRLINK=REC_ERR,-
                                      02C6  3903                          P1=#T_REC_END
                    00001D2E'8F   DD  02C6                         PUSHL   #T_REC_END
                    00001EB4'EF   DF  02CC                         PUSHAL  REC_ERR
                    00001888'EF   DF  02D2                         PUSHAL  T_SLF_DIAG_ERR
                             00   DD  02D8                         PUSHL   #0
                             02   DD  02DA                         PUSHL   #$ER
                                      02DC       ;;; TEST 9, SUBTEST 2, ERROR 2
          000100D0 9F        05   FB  02DC                         CALLS   #$$M, a#DS$ERRHARD
                                      02E3  3904          $DS_BNCOMPLETE  5$
                          03 50   E9  02E3                         BLBC    R0, 5$
                                      02E6  3905          $DS_ABORT       TEST
                             50   D4  02E6                         CLRL    R0                   ; SEND A WARNING TO THE SUPERVISOR
                             04   02E8                             RET                          ; TERMINATE TEST
                                      02E9  3906 5$:      $DS_ABORT
```

```
              00010020 9F   6C   FA  02E9                       CALLG   (AP), @#DS$ABORT
                                     02F0  3907
                       5C   0E   D0  02F0  3908 6$:     MOVL            #14,R12                 ; load number of bytes to check
                                     02F3  3909         $DS_BREAK
              00010058 9F   6E   FA  02F3                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16  02FA  3910         JSB     CHECK_PACKET            ; go check end packet
                                     0300  3911         $DS_BNCOMPLETE  7$              ; br = no error
                         21 50   E9  0300                       BLBC    R0, 7$
                                     0303  3912         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_SLF_DIAG_ERR,PRLINK=DATA_ERR_END
                   00001E9D'EF   DF  0303                       PUSHAL  DATA_ERR_END
                   00001888'EF   DF  0309                       PUSHAL  T_SLF_DIAG_ERR
                            00   DD  030F                       PUSHL   #0
                            03   DD  0311                       PUSHL   #$ER
                                     0313       ;;; TEST 9, SUBTEST 2, ERROR 3
              000100D0 9F   04   FB  0313                       CALLS   #$$M, @#DS$ERRHARD
                                     031A  3913         $DS_BCOMPLETE   7$              ; continue if only data error
                         07 50   E8  031A                       BLBS    R0, 7$
                                     031D  3914         $DS_ABORT
              00010020 9F   6C   FA  031D                       CALLG   (AP), @#DS$ABORT
                                     0324  3915 7$:
                                     0324  3916
                                     0324  3917         $DS_ENDSUB
                                     0324       T9_S2_X:
     00010038 9F   0000000C'EF   FA  0324                       CALLG   $S$, @#DS$ENDSUB
                                     032F  3918
                                     032F  3919         $DS_PAGE
```

```
032F                 .SBTTL  SUBTEST 9.3:   NOP
032F       ;+
032F       ; SUBTEST DESCRIPTION:
032F       ;
032F       ;       This subtest sends the TU58 a command packet to perform a no
032F       ;       operation. While sending each byte of the packet, a timer is set which
032F       ;       will time out if the ready bit in the transmitter processor register
032F       ;       does not set within the alloted time. The same is done for the done
032F       ;       bit in the receiver processor register while reading a byte from the
032F       ;       TU58. When this is completed successfully, the returned end packet is
032F       ;       compared to an expected end packet. If a time out occurs while expecting
032F       ;       a byte transfer or bad data is encountered an error will be reported.
032F       ;
032F       ;
032F       ; SUBTEST STEPS:
032F       ;
032F       ;       SUBTEST NOP
032F       ;       :  Clear buffer
032F       ;       :  Point to no operation command packet
032F       ;       :  REPEAT
032F       ;       :  :  Send command packet byte at a time
032F       ;       :  :  Set timer
032F       ;       :  :  IF timer expired AND no response
032F       ;       :  :  :  THEN
032F       ;       :  :  :  Report no response error
032F       ;       :  :  :  Exit test
032F       ;       :  :  ENDIF
032F       ;       :  : UNTIL all bytes sent
032F       ;       :  ENDREPEAT
032F       ;       :  REPEAT
032F       ;       :  :  Set timer
032F       ;       :  :  IF timer expired AND done set
032F       ;       :  :  :  THEN
032F       ;       :  :  :  :  Read byte
032F       ;       :  :  ENDIF
032F       ;       :  : UNTIL timer expired AND done clear
032F       ;       :  ENDREPEAT
032F       ;       :  IF expected byte count NEQ actual byte count
032F       ;       :  :  THEN
032F       ;       :  :  :  Report error
032F       ;       :  :  :  Exit test
032F       ;       :  ENDIF
032F       ;       :  Point to expected and actual end packet
032F       ;       :  REPEAT
032F       ;       :  :  IF expected data NEQ received data
032F       ;       :  :  :  THEN
032F       ;       :  :  :  Report data error
032F       ;       :  :  :  Exit test
032F       ;       :  :  ENDIF
032F       ;       :  : UNTIL all data checked
032F       ;       :  ENDREPEAT
032F       ;       ENDSUBTEST NOP
032F       ;
032F       ;
032F       ;
```

```
                                   032F  3976 ;.page
                                   032F  3977 ;-
                                   032F  3978          $DS_BGNSUB
                                   032F       T9_S3::
     00010030 9F   00000018'EF   FA 032F                        CALLG   $S$, a#DS$BGNSUB
                                   033A  3979
                                   033A  3980
                   00000000'EF   94 033A  3981          CLRB    B_FLAGS                   ; initialize flags
          58       00000000'FF   9E 0340  3982          MOVAB   aA_BUFFER_BEGIN,R8        ; load buffer starting address
          59       00000400 8F   D0 0347  3983          MOVL    #1024,R9                  ; do both pages
                   00002134'EF   16 034E  3984          JSB     CLEAR_BUFFER              ; go clear buffers
          54       FFFB6C20 8F   D0 0354  3985          MOVL    #DELAY_TIME1,R4           ; load send packet delay time
          55       00000032'EF   9E 035B  3986          MOVAB   B_NOP_CMD,R5              ; point to init command packet
                        57   0E  D0 0362  3987          MOVL    #14,R7                    ; load transfer byte count
                   0000203D'EF   16 0365  3988          JSB     SEND_TU58                 ; send command packet
       24 00000000'EF    00  E1 036B  3989          BBC     #V_CLOCK_FLAG,B_FLAGS,2$ ; br = no timeout error
                                   0373  3990          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_NOP_ERR,PRLINK=NO_READY
                   00001BEB'EF   DF 0373                        PUSHAL  NO_READY
                   000018AD'EF   DF 0379                        PUSHAL  T_NOP_ERR
                            00   DD 037F                        PUSHL   #0
                            01   DD 0381                        PUSHL   #$ER
                                   0383       ;;; TEST 9, SUBTEST 3, ERROR 1
          000100D0 9F        04   FB 0383                        CALLS   #$$M, a#DS$ERRHARD
                                   038A  3991          $DS_BNCOMPLETE  1$
                         03 50   E9 038A                        BLBC    R0, 1$
                                   038D  3992          $DS_ABORT       TEST
                            50   D4 038D                        CLRL    R0                ; SEND A WARNING TO THE SUPERVISOR
                            04      038F                        RET                       ; TERMINATE TEST
                                   0390  3993 1$:      $DS_ABORT
          00010020 9F        6C   FA 0390                        CALLG   (AP), a#DS$ABORT
                                   0397  3994
          54    FFFB6C20 8F   D0 0397  3995 2$:      MOVL    #DELAY_TIME1,R4           ; load receive packet delay time
          55    00000000'FF   9E 039E  3996          MOVAB   aA_BUFFER_BEGIN,R5        ; load buffer address to store data
              57    55   0E   C1 03A5  3997          ADDL3   #14,R5,R7                 ; load last expected address of data
                00002083'EF   16 03A9  3998          JSB     RECEIVE_TU58              ; go get data from TU58
          5A    000005D4'EF   9E 03AF  3999          MOVAB   B_END_PACKET_1,R10        ; load address of expected end packet
          5B    00000000'FF   9E 03B6  4000          MOVAB   aA_BUFFER_BEGIN,R11       ; load address of actual end packet
                      57    55   D1 03BD  4001          CMPL    R5,R7                     ; check for expected number of bytes
                            31   13 03C0  4002          BEQL    6$                        ; br = correct nbumber of bytes
          00000000'EF       08   88 03C2  4003          BISB2   #M_END_FLAG,B_FLAGS       ; set end error flag
                                   03C9  4004 4$:      $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_NOP_ERR,PRLINK=REC_ERR,-
                                   03C9  4005                          P1=#T_REC_END
                   00001D2E'8F   DD 03C9                        PUSHL   #T_REC_END
                   00001EB4'EF   DF 03CF                        PUSHAL  REC_ERR
                   000018AD'EF   DF 03D5                        PUSHAL  T_NOP_ERR
                            00   DD 03DB                        PUSHL   #0
                            02   DD 03DD                        PUSHL   #$ER
                                   03DF       ;;; TEST 9, SUBTEST 3, ERROR 2
          000100D0 9F        05   FB 03DF                        CALLS   #$$M, a#DS$ERRHARD
                                   03E6  4006          $DS_BNCOMPLETE  5$
                         03 50   E9 03E6                        BLBC    R0, 5$
                                   03E9  4007          $DS_ABORT       TEST
                            50   D4 03E9                        CLRL    R0                ; SEND A WARNING TO THE SUPERVISOR
                            04      03EB                        RET                       ; TERMINATE TEST
                                   03EC  4008 5$:      $DS_ABORT
```

```
            00010020 9F    6C    FA  03EC                        CALLG   (AP), @#DS$ABORT
                                     03F3  4009
                     5C    0E    D0  03F3  4010 6$:     MOVL    #14,R12                 ; load number of bytes to check
                                     03F6  4011         $DS_BREAK
            00010058 9F    6E    FA  03F6                        CALLG   (SP), @#DS$BREAK
                 0000213B'EF     16  03FD  4012         JSB     CHECK_PACKET            ; go check end packet
                                     0403  4013         $DS_BNCOMPLETE  7$              ; br = no error
                       21 50     E9  0403                        BLBC    R0, 7$
                                     0406  4014         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_NOP_ERR,PRLINK=DATA_ERR_END
                 00001E9D'EF     DF  0406                        PUSHAL  DATA_ERR_END
                 000018AD'EF     DF  040C                        PUSHAL  T_NOP_ERR
                           00    DD  0412                        PUSHL   #0
                           03    DD  0414                        PUSHL   #$ER
                                     0416       ;;; TEST 9, SUBTEST 3, ERROR 3
            000100D0 9F    04    FB  0416                        CALLS   #$$M, @#DS$ERRHARD
                                     041D  4015         $DS_BCOMPLETE   7$              ; continue if only data error
                       07 50     E8  041D                        BLBS    R0, 7$
                                     0420  4016         $DS_ABORT
            00010020 9F    6C    FA  0420                        CALLG   (AP), @#DS$ABORT
                                     0427  4017 7$:
                                     0427  4018
                                     0427  4019         $DS_ENDSUB
                                     0427       T9_S3_X:
   00010038 9F   00000018'EF     FA  0427                        CALLG   $S$, @#DS$ENDSUB
                                     0432  4020         $DS_CLRVEC_S    VECTOR=#SCB$L_TIMER
                 000000C0 8F     DD  0432                        PUSHL   #SCB$L_TIMER
            00010168 9F    01    FB  0438                        CALLS   #1, @#DS$CLRVEC
                                     043F  4021
                                     043F  4022         $DS_ENDTEST
                     50    01    D0  043F                        MOVL    #1, R0          ; NORMAL EXIT
                                     0442       TEST_009_X::
            00010058 9F    6E    FA  0442                        CALLG   (SP), @#DS$BREAK
                                 04  0449                        RET                     ; RETURN TO TEST SEQUENCER
                                     044A  4023
                                     044A  4024         $DS_PAGE
```

```
                                044A              .SBTTL  TEST 10:  TU58 EXERCISER READ/WRITE
                            00000000                      .PSECT  TEST_010, PAGE, NOWRT
                                001C
                                001C  4027 ;++
                                001C  4028 ; Test description:
                                001C  4029 ;
                                001C  4030 ;          This test exercises the TU58 on the VAX-11/750 by performing
                                001C  4031 ;          position, read (normal and two versions of modified) and write
                                001C  4032 ;          (normal and two versions of modified) commands. Zero padding of
                                001C  4033 ;          blocks and records are also checked for write commands. All end and
                                001C  4034 ;          data packets are checked for errors. Timers are setup to time out if
                                001C  4035 ;          a byte transaction between the host and TU58 is not completed within
                                001C  4036 ;          a reasonable amount of time. If any errors are detected, they are
                                001C  4037 ;          reported with all available information pertaining to the failure.
                                001C  4038 ;
                                001C  4039 ;
                                001C  4040 ; Assumptions:
                                001C  4041 ;
                                001C  4042 ;          This test assumes the scratch tape cartridge used is in good
                                001C  4043 ;          condition. If a marginal tape is used bits may be dropped during the
                                001C  4044 ;          read modified test, which reads with an increased threshold and will
                                001C  4045 ;          consequently report false data errors. Any tape may be used, but
                                001C  4046 ;          existing data will be destroyed. Before the tape is written, the first
                                001C  4047 ;          directory will be read. If the directory read is not 'scrat.ch', the
                                001C  4048 ;          operator will be prompted to either overwrite the tape or abort this
                                001C  4049 ;          test. If overwriting the tape is chosen, an attempt to write
                                001C  4050 ;           scrat.ch" in the first directory area will be made. If an operational
                                001C  4051 ;          error occurs it will be reported, but error detection in this subtest
                                001C  4052 ;          is limited and the primary function is to give the operator an
                                001C  4053 ;          opportunity to remove a tape containing valuable data from the drive
                                001C  4054 ;          and inserting a scratch tape before it is destroyed.
                                001C  4055 ;
                                001C  4056 ;
                                001C  4057 ;.page
                                001C  4058 ;--
                                001C  4059          $DS_BGNTEST      <DEFAULT,TU58,ALL>
                                001C       DATA_010:
                    00000000    001C                      .LONG    0                ; TEST ARGUMENT TABLE TERMINATOR
                                0020       TEST_010::
                        0000    0020                      .WORD    ^M<>             ; ENTRY MASK
                                0022  4060
                                0022  4061
                                0022  4062          ; Set Up
                                0022  4063
                                0022  4064          $DS_SETVEC_S     VECTOR=#SCB$L_TIMER,SRVADR=TIMER_SERVICE
                   00      DD   0022                      PUSHL    #0
          00002150'EF      DF   0024                      PUSHAL   TIMER_SERVICE
          000000C0 8F      DD   002A                      PUSHL    #SCB$L_TIMER
    00010160 9F    03      FB   0030                      CALLS    #3, @#DS$SETVEC
                                0037  4065
          58   00000000'FF     9E   0037  4066 1$:    MOVAB    @A_BUFFER_BEGIN,R8       ; load buffer starting address
00000013'EF    58   0000042E 8F    C1   003E  4067        ADDL3    #1070,R8,BUF_LIM         ; construct upper buffer limit
          59   00000400 8F     D0   004A  4068        MOVL     #1024,R9                 ; do both pages
          00002134'EF      16   0051  4069            JSB      CLEAR_BUFFER             ; go clear buffers
                                0057  4070
                   00000000'EF   94  0057  4071         CLRB    B_FLAGS                 ; init flags
                   00000001'EF   94  005D  4072         CLRB    B_FLAGS2                ; init flags
                   00001F59'EF   16  0063  4073 2$:     JSB     INIT_SEQUENCE           ; initialize drive
              5A   00000023'EF   9E  0069  4074         MOVAB   B_CONTINUE,R10          ; load expected data address
              5B   00000000'FF   9E  0070  4075         MOVAB   @A_BUFFER_BEGIN,R11     ; load actual data address
           2C 00000000'EF   04   E1  0077  4076         BBC     #V_SEND_FLAG,B_FLAGS,5$ ; br = no send error
           DC 00000001'EF   01   E3  007F  4077         BBCS    #V_REPEAT_SEND,B_FLAGS2,2$ ; br = first try and set flag
                                     0087  4078         $DS_ERRHARD_S  ,UNIT=#0,MSGADR=T_INIT1_ERR,PRLINK=NO_READY
                   00001BEB'EF   DF  0087                       PUSHAL  NO_READY
                   0000183B'EF   DF  008D                       PUSHAL  T_INIT1_ERR
                            00   DD  0093                       PUSHL   #0
                            01   DD  0095                       PUSHL   #$ER
                                     0097       ;;; TEST 10, SUBTEST 0, ERROR 1
              000100D0 9F   04   FB  0097                       CALLS   #$$M, @#DS$ERRHARD
                                     009E  4079         $DS_BNCOMPLETE  3$
                         03 50   E9  009E                       BLBC    R0, 3$
                                     00A1  4080         $DS_ABORT       TEST
                            50   D4  00A1                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  00A3                       RET                     ; TERMINATE TEST
                                     00A4  4081 3$:     $DS_ABORT
              00010020 9F   6C   FA  00A4                       CALLG   (AP), @#DS$ABORT
                                     00AB  4082
           32 00000000'EF   01   E1  00AB  4083 5$:     BBC     #V_RECEIVE_FLAG,B_FLAGS,9$ ; br = no receive error
           A8 00000001'EF   01   E3  00B3  4084         BBCS    #V_REPEAT_SEND,B_FLAGS2,2$ ; br = first try and set flag
                                     00BB  4085 7$:     $DS_ERRHARD_S  ,UNIT=#0,MSGADR=T_INIT1_ERR,PRLINK=REC_ERR,-
                                     00BB  4086                         P1=#T_REC_CONT
                   00001D47'8F   DD  00BB                       PUSHL   #T_REC_CONT
                   00001EB4'EF   DF  00C1                       PUSHAL  REC_ERR
                   0000183B'EF   DF  00C7                       PUSHAL  T_INIT1_ERR
                            00   DD  00CD                       PUSHL   #0
                            02   DD  00CF                       PUSHL   #$ER
                                     00D1       ;;; TEST 10, SUBTEST 0, ERROR 2
              000100D0 9F   05   FB  00D1                       CALLS   #$$M, @#DS$ERRHARD
                                     00D8  4087         $DS_BNCOMPLETE  8$
                         03 50   E9  00D8                       BLBC    R0, 8$
                                     00DB  4088         $DS_ABORT       TEST
                            50   D4  00DB                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  00DD                       RET                     ; TERMINATE TEST
                                     00DE  4089 8$:     $DS_ABORT
              00010020 9F   6C   FA  00DE                       CALLG   (AP), @#DS$ABORT
                                     00E5  4090
                       5C   01   D0  00E5  4091 9$:     MOVL    #1,R12                  ; load number of bytes to check
                                     00E8  4092         $DS_BREAK
              00010058 9F   6E   FA  00E8                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16  00EF  4093         JSB     CHECK_PACKET            ; go check continue packet
                                     00F5  4094         $DS_BNCOMPLETE  12$
                         39 50   E9  00F5                       BLBC    R0, 12$
           0A 00000001'EF   01   E0  00F8  4095         BBS     #V_REPEAT_SEND,B_FLAGS2,10$ ; br = first try and set flag
              00000001'EF   02   88  0100  4096         BISB2   #M_REPEAT_SEND,B_FLAGS2
                          FF59   31  0107  4097         BRW     2$
                       52   50   D0  010A  4098 10$:    MOVL    R0,R2                   ; save code for error report
                                     010D  4099         $DS_ERRHARD_S  ,UNIT=#0,MSGADR=T_INIT1_ERR,PRLINK=DATA_ERR_CONT
                   00001E4D'EF   DF  010D                       PUSHAL  DATA_ERR_CONT
                   0000183B'EF   DF  0113                       PUSHAL  T_INIT1_ERR
                            00   DD  0119                       PUSHL   #0
```

```
                     03   DD  011B                      PUSHL   #$ER
                              011D        ;;; TEST 10, SUBTEST 0, ERROR 3
        000100D0 9F  04   FB  011D                      CALLS   #$$M, @#DS$ERRHARD
                              0124  4100        $DS_BNCOMPLETE  11$
                  03 50   E9  0124                      BLBC    R0, 11$
                              0127  4101        $DS_ABORT       TEST
                     50   D4  0127                      CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                          04  0129                      RET                     ; TERMINATE TEST
                              012A  4102 11$:   $DS_ABORT
        00010020 9F  6C   FA  012A                      CALLG   (AP), @#DS$ABORT
                              0131  4103 12$:
                              0131  4104        $DS_PAGE
```

```
0131                 .SBTTL  SUBTEST 10.1:   READ DIRECTORY
0131       ;+
0131       ; SUBTEST DESCRIPTION:
0131       ;
0131       ;       This subtest sends the TU58 a command packet to read the first entry
0131       ;       in the directory area of the tape. A timer is set which will
0131       ;       time out if the ready bit in the transmitter processor register does
0131       ;       not set within the allotted time. A second timer is set which will
0131       ;       time out if the done bit in the receiver processor register does not
0131       ;       set within the allotted amount of time. If either of these timers
0131       ;       expires while expecting a data transfer an error will be reported
0131       ;       stating the conditions existing at the time of failure.
0131       ;       The returned data packet is examined to see if the directory on the
0131       ;       tape is 'scrat.ch". If it is a scratch tape testing continues; if not,
0131       ;       the operator will be prompted to either abort the TU58 exerciser test
0131       ;       or overwrite the tape in the drive. Before entering the appropriate
0131       ;       response on the TTY, the tape cartridge may be replaced with a scratch
0131       ;       tape. In the case of an overwrite, the TU58 will be sent a write
0131       ;       command  packet to write "scrat.ch" in the directory.  Timers are set
0131       ;       which will time out if the appropriate bits do not set within allotted
0131       ;       times. If an error occurs all appropriate information will be
0131       ;       reported. This subtest is not primarily used for error detection, if an
0131       ;       error is encountered, testing should continue with the next subtest
0131       ;       for diagnosis.
0131       ;
0131       ;
0131       ; SUBTEST STEPS:
0131       ;
0131       ;       SUBTEST READ DIRECTORY
0131       ;       :  Clear buffer
0131       ;       :  Point to read command packet
0131       ;       :  REPEAT
0131       ;       :  :  Send byte of command packet
0131       ;       :  :  Set timer
0131       ;       :  :  IF no response AND timer expired
0131       ;       :  :  :  THEN
0131       ;       :  :  :  Report no response error
0131       ;       :  :  :  Exit test
0131       ;       :  :  ENDIF
0131       ;       :  : UNTIL all bytes sent
0131       ;       :  ENDREPEAT
0131       ;       :  REPEAT
0131       ;       :  :  Set timer
0131       ;       :  :  IF timer expired AND done set
0131       ;       :  :  :  THEN
0131       ;       :  :  :  :  Read byte
0131       ;       :  :  ENDIF
0131       ;       :  : UNTIL timer expired AND done clear
0131       ;       :  ENDREPEAT
0131       ;       :  IF expected byte count NEQ actual byte count
0131       ;       :  :  THEN
0131       ;       :  :  :  Report error
0131       ;       :  :  :  Exit test
0131       ;       :  ENDIF
0131       ;       :  Examine data for first directory on tape
```

```
0131 ;    : IF directory NEQ scrat.ch
0131 ;    : : THEN
0131 ;    : : WHILE write directory flag EQL zero DO
0131 ;    : : : Prompt operator to overwrite directory
0131 ;    : : : IF response EQL no
0131 ;    : : : : THEN
0131 ;    : : : : Exit test
0131 ;    : : : ENDIF
0131 ;    : : : IF response EQL yes
0131 ;    : : : : THEN
0131 ;    : : : : Set write directory flag
0131 ;    : : : : Point to write directory command packet
0131 ;    : : : : REPEAT
0131 ;    : : : : : Send command packet byte at a time
0131 ;    : : : : : Set timer
0131 ;    : : : : : IF timer expired AND no response
0131 ;    : : : : : : THEN
0131 ;    : : : : : : Report no response error
0131 ;    : : : : : : Exit test
0131 ;    : : : : : ENDIF
0131 ;    : : : : :UNTIL all bytes sent
0131 ;    : : : : ENDREPEAT
0131 ;    : : : : Wait for continue
0131 ;    : : : : Set timer
0131 ;    : : : : IF timer expired AND no response
0131 ;    : : : : : THEN
0131 ;    : : : : : Report no response error
0131 ;    : : : : : Exit test
0131 ;    : : : : ENDIF
0131 ;    : : : : Point to data packet with  scrat.ch' directory
0131 ;    : : : : REPEAT
0131 ;    : : : : : Send data packet byte at a time
0131 ;    : : : : : Set timer
0131 ;    : : : : : IF timer expired AND no response
0131 ;    : : : : : : THEN
0131 ;    : : : : : : Report no response error
0131 ;    : : : : : : Exit test
0131 ;    : : : : : ENDIF
0131 ;    : : : : :UNTIL all bytes sent
0131 ;    : : : : ENDREPEAT
0131 ;    : : : : REPEAT
0131 ;    : : : : : Read end packet into buffer byte at a time
0131 ;    : : : : : Set timer
0131 ;    : : : : : IF timer expired AND no response
0131 ;    : : : : : : THEN
0131 ;    : : : : : : Report no response error
0131 ;    : : : : : : Exit test
0131 ;    : : : : : ENDIF
0131 ;    : : : : : :UNTIL all bytes read
0131 ;    : : : : ENDREPEAT
0131 ;    : : : : Point to expected and received end packets
0131 ;    : : : : REPEAT
0131 ;    : : : : : IF expected data NEQ received data
0131 ;    : : : : : : THEN
0131 ;    : : : : : : Report data error
```

```
                                      0131  4216 ;        :  :  :  :  :  :  Exit subtest
                                      0131  4217 ;        :  :  :  :  :  ENDIF
                                      0131  4218 ;        :  :  :  :  : UNTIL all bytes checked
                                      0131  4219 ;        :  :  :  :  ENDREPEAT
                                      0131  4220 ;        :  :  :  ENDIF
                                      0131  4221 ;        :  :  ENDWHILE
                                      0131  4222 ;        :  ENDIF
                                      0131  4223 ;        ENDSUBTEST READ DIRECTORY
                                      0131  4224 ;
                                      0131  4225 ;
                                      0131  4226 ;
                                      0131  4227 ;.page
                                      0131  4228 ;-
                                      0131  4229
                                      0131  4230
                                      0131  4231          $DS_BGNSUB
                                      0131       T10_S1::
      00010030 9F    00000024'EF   FA 0131                        CALLG   $S$, @#DS$BGNSUB
                                      013C  4232                                                ; clear buffers
                     00000000'EF   94 013C  4233          CLRB    B_FLAGS                       ; initialize flags
                58   00000000'FF   9E 0142  4234          MOVAB   @A_BUFFER_BEGIN,R8            ; load buffer starting address
                59   00000400 8F   D0 0149  4235          MOVL    #1024,R9                      ; do two pages
                     00002134'EF   16 0150  4236          JSB     CLEAR_BUFFER                  ; go clear buffers
                                      0156  4237                                                ; send read command packet
                54   FF7E4CE0 8F   D0 0156  4238          MOVL    #DELAY_TIME7,R4               ; load send packet delay time
                55   0000004E'EF   9E 015D  4239          MOVAB   B_READ_1_CMD,R5               ; point to read command packet
                          57  0E   D0 0164  4240          MOVL    #14,R7                        ; load transfer byte count
                     0000203D'EF   16 0167  4241          JSB     SEND_TU58                     ; send command packet
            24 00000000'EF    00   E1 016D  4242          BBC     #V_CLOCK_FLAG,B_FLAGS,2$ ; br = no timeout error
                                      0175  4243          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=NO_READY
                     00001BEB'EF   DF 0175                        PUSHAL  NO_READY
                     00001931'EF   DF 017B                        PUSHAL  T_RD_REC_ERR
                              00   DD 0181                        PUSHL   #0
                              01   DD 0183                        PUSHL   #$ER
                                      0185       ;;; TEST 10, SUBTEST 1, ERROR 1
           000100D0 9F        04   FB 0185                        CALLS   #$$M, @#DS$ERRHARD
                                      018C  4244          $DS_BNCOMPLETE  1$
                           03 50   E9 018C                        BLBC    R0, 1$
                                      018F  4245          $DS_ABORT       TEST
                              50   D4 018F                        CLRL    R0                    ; SEND A WARNING TO THE SUPERVISOR
                              04   04 0191                        RET                           ; TERMINATE TEST
                                      0192  4246 1$:      $DS_ABORT
           00010020 9F        6C   FA 0192                        CALLG   (AP), @#DS$ABORT
                                      0199  4247                                                ; read data and end packet
                54   FAEF00C0 8F   D0 0199  4248 2$:      MOVL    #DELAY_TIME8,R4               ; load receive packet delay time
                55   00000000'FF   9E 01A0  4249          MOVAB   @A_BUFFER_BEGIN,R5            ; load buffer address to store data
           57   55   00000092 8F   C1 01A7  4250          ADDL3   #146,R5,R7                    ; load last expected address of data
                     00002083'EF   16 01AF  4251          JSB     RECEIVE_TU58                  ; go get data from TU58
                          57  55   D1 01B5  4252          CMPL    R5,R7                         ; check for correct number of bytes
                              24   13 01B8  4253          BEQL    6$                            ; br = correct number of bytes
                                      01BA  4254          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=MSG_1
                     00001AD6'EF   DF 01BA                        PUSHAL  MSG_1
                     00001931'EF   DF 01C0                        PUSHAL  T_RD_REC_ERR
                              00   DD 01C6                        PUSHL   #0
                              02   DD 01C8                        PUSHL   #$ER
```

```
                             01CA        ;;; TEST 10, SUBTEST 1, ERROR 2
        000100D0 9F     04   FB  01CA                CALLS    #$$M, @#DS$ERRHARD
                             01D1  4255        $DS_BNCOMPLETE  4$
                03 50   E9   01D1                    BLBC     R0, 4$
                             01D4  4256        $DS_ABORT       TEST
                   50   D4   01D4                    CLRL     R0                  ; SEND A WARNING TO THE SUPERVISOR
                        04   01D6                    RET                          ; TERMINATE TEST
                             01D7  4257
                             01D7  4258 4$:    $DS_ABORT
        00010020 9F     6C   FA  01D7                CALLG    (AP), @#DS$ABORT
                             01DE  4259                                           ; check end packet
        5A    000005E2'EF   9E   01DE  4260 6$:    MOVAB    B_END_PACKET_2,R10     ; load address of expected end packet
        5B    00000000'FF   9E   01E5  4261        MOVAB    @A_BUFFER_BEGIN,R11    ; load address of actual end packet
        5B    00000084 8F   C0   01EC  4262        ADDL2    #132,R11               ; skip over data packet in buffer
                  5C   0E   D0   01F3  4263        MOVL     #14,R12                ; load number of bytes to check
                             01F6  4264        $DS_BREAK
        00010058 9F     6E   FA  01F6                CALLG    (SP), @#DS$BREAK
             0000213B'EF    16   01FD  4265        JSB      CHECK_PACKET           ; go check end packet
                             0203  4266        $DS_BNCOMPLETE  7$                 ; br = no error
                21 50   E9   0203                    BLBC     R0, 7$
                             0206  4267        $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=DATA_ERR_END
             00001E9D'EF    DF   0206                PUSHAL   DATA_ERR_END
             00001931'EF    DF   020C                PUSHAL   T_RD_REC_ERR
                        00   DD   0212               PUSHL    #0
                        03   DD   0214               PUSHL    #$ER
                             0216        ;;; TEST 10, SUBTEST 1, ERROR 3
        000100D0 9F     04   FB  0216                CALLS    #$$M, @#DS$ERRHARD
                             021D  4268        $DS_BCOMPLETE   7$                 ; br = data error, but continue
                07 50   E8   021D                    BLBS     R0, 7$
                             0220  4269        $DS_ABORT
        00010020 9F     6C   FA  0220                CALLG    (AP), @#DS$ABORT
                             0227  4270                                           ; check first directory entry
        5A    000006EC'EF   9E   0227  4271 7$:    MOVAB    B_DATA_PKT_1,R10       ; load address of expected data packet
                  5A   0E   C0   022E  4272        ADDL2    #14,R10                ; point to expected first directory entry
        5B    00000000'FF   9E   0231  4273        MOVAB    @A_BUFFER_BEGIN,R11    ; load address of actual data packet
                  5B   0E   C0   0238  4274        ADDL2    #14,R11                ; point to actual first directory entry
                  5C   06   D0   023B  4275        MOVL     #6,R12                 ; do only directory entry
                             023E  4276        $DS_BREAK
        00010058 9F     6E   FA  023E                CALLG    (SP), @#DS$BREAK
             0000213B'EF    16   0245  4277        JSB      CHECK_PACKET           ; go check data packet
                             024B  4278        $DS_BCOMPLETE   8$                 ; br =  unknown tape
                03 50   E8   024B                    BLBS     R0, 8$
                             024E  4279        $DS_EXIT        SUB                ; exit subtest if scratch tape
                  01E5  31   024E                    BRW      T10_S1_X            ; EXIT SUBTEST 1
                             0251  4280 8$:    $DS_ASKLGCL_S   T_OVERWRITE,B_RESPONSE,,9$,0,
                        00   DD   0251               PUSHL    #0
                        00   DD   0253               PUSHL    #PAR$_NO
                        00   DD   0255               PUSHL    #0
             00000278'EF    DF   0257                PUSHAL   9$
                        00   DD   025D               PUSHL    #0
             00000012'EF    DF   025F                PUSHAL   B_RESPONSE
             00001B56'EF    DF   0265                PUSHAL   T_OVERWRITE
        00010098 9F     07   FB  026B                CALLS    #7, @#DS$ASKLGCL
                             0272  4281        $DS_EXIT        TEST               ; exit test if negative response
                  50   01   D0   0272                    MOVL     #1,R0               ; NOT AN ERROR
```

```
                 1EC2   31  0275                       BRW     TEST_010_X           ; EXIT TEST 10
                            0278  4282                                               ; positive response, write directory
      54   FF7E4CE0 8F  D0  0278  4283 9$:      MOVL    #DELAY_TIME7,R4          ; load send packet delay time
      55   0000005C'EF  9E  027F  4284          MOVAB   B_WRITE_1_CMD,R5         ; point to write command packet
               57   0E  D0  0286  4285          MOVL    #14,R7                   ; load transfer byte count
         0000203D'EF    16  0289  4286          JSB     SEND_TU58                ; send command packet
   24 00000000'EF   00  E1  028F  4287          BBC     #V_CLOCK_FLAG,B_FLAGS,20$ ; br = no timeout error
                            0297  4288          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=NO_READY
         00001BEB'EF    DF  0297                       PUSHAL  NO_READY
         0000190D'EF    DF  029D                       PUSHAL  T_WR_REC_ERR
                   00   DD  02A3                       PUSHL   #0
                   04   DD  02A5                       PUSHL   #$ER
                            02A7       ;;; TEST 10, SUBTEST 1, ERROR 4
      000100D0 9F   04  FB  02A7                       CALLS   #$$M, @#DS$ERRHARD
                            02AE  4289          $DS_BNCOMPLETE  10$
                03 50   E9  02AE                       BLBC    R0, 10$
                            02B1  4290          $DS_ABORT      TEST
                   50   D4  02B1                       CLRL    R0                   ; SEND A WARNING TO THE SUPERVISOR
                   04       02B3                       RET                          ; TERMINATE TEST
                            02B4  4291 10$:     $DS_ABORT
      00010020 9F   6C  FA  02B4                       CALLG   (AP), @#DS$ABORT
                            02BB  4292                                               ; read continue
      54   FFF3CB00 8F  D0  02BB  4293 20$:     MOVL    #DELAY_TIME3,R4          ; load receive packet delay time
      55   00000000'FF  9E  02C2  4294          MOVAB   @A_BUFFER_BEGIN,R5       ; load buffer address to store data
      55   00000096 8F  C0  02C9  4295          ADDL2   #150,R5                  ; get above previous data
        57   55     01  C1  02D0  4296          ADDL3   #1,R5,R7                 ; load last expected address of data
         00002083'EF    16  02D4  4297          JSB     RECEIVE_TU58             ; go get data from TU58
      5A   00000023'EF  9E  02DA  4298          MOVAB   B_CONTINUE,R10           ; load address of expected data
      5B   00000000'FF  9E  02E1  4299          MOVAB   @A_BUFFER_BEGIN,R11      ; load address of buffer
      5B   00000096 8F  C0  02E8  4300          ADDL    #150,R11                 ; point to address of actual data
               57   55  D1  02EF  4301          CMPL    R5,R7                    ; check for correct number of bytes
                   2A   13  02F2  4302          BEQL    60$                      ; br = correct number of bytes
                            02F4  4303 25$:     $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK-REC_ERR,-
                            02F4  4304                           P1=#T_REC_CONT
         00001D47'8F    DD  02F4                       PUSHL   #T_REC_CONT
         00001EB4'EF    DF  02FA                       PUSHAL  REC_ERR
         0000190D'EF    DF  0300                       PUSHAL  T_WR_REC_ERR
                   00   DD  0306                       PUSHL   #0
                   05   DD  0308                       PUSHL   #$ER
                            030A       ;;; TEST 10, SUBTEST 1, ERROR 5
      000100D0 9F   05  FB  030A                       CALLS   #$$M, @#DS$ERRHARD
                            0311  4305          $DS_BNCOMPLETE  30$
                03 50   E9  0311                       BLBC    R0, 30$
                            0314  4306          $DS_ABORT      TEST
                   50   D4  0314                       CLRL    R0                   ; SEND A WARNING TO THE SUPERVISOR
                   04       0316                       RET                          ; TERMINATE TEST
                            0317  4307
                            0317  4308 30$:     $DS_ABORT
      00010020 9F   6C  FA  0317                       CALLG   (AP), @#DS$ABORT
                            031E  4309                                               ; check continue packet
               5C   01  D0  031E  4310 60$:     MOVL    #1,R12                   ; load number of bytes to check
                            0321  4311          $DS_BREAK
      00010058 9F   6E  FA  0321                       CALLG   (SP), @#DS$BREAK
         0000213B'EF    16  0328  4312          JSB     CHECK_PACKET             ; go check continue
                            032E  4313          $DS_BNCOMPLETE  75$              ; br = no error
```

```
                         24 50   E9  032E                           BLBC    R0, 75$
                                     0331  4314         $DS_ERRHARD_S  ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=DATA_ERR_DATA
                   00001E75'EF   DF  0331                           PUSHAL  DATA_ERR_DATA
                   0000190D'EF   DF  0337                           PUSHAL  T_WR_REC_ERR
                            00   DD  033D                           PUSHL   #0
                            06   DD  033F                           PUSHL   #$ER
                                     0341       ;;; TEST 10, SUBTEST 1, ERROR 6
              000100D0 9F   04   FB  0341                           CALLS   #$$M, @#DS$ERRHARD
                                     0348  4315         $DS_BNCOMPLETE  70$             ; br = error, but continue
                         03 50   E9  0348                           BLBC    R0, 70$
                                     034B  4316         $DS_ABORT       TEST
                            50   D4  034B                           CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  034D                           RET                     ; TERMINATE TEST
                                     034E  4317 70$:    $DS_ABORT
              00010020 9F   6C   FA  034E                           CALLG   (AP), @#DS$ABORT
                                     0355  4318                                         ; send data packet
              54    FFFB6C20 8F   D0  0355  4319 75$:    MOVL    #DELAY_TIME1,R4         ; load send packet delay time
              55    000006EC'EF   9E  035C  4320         MOVAB   B_DATA_PKT_1,R5         ; point to data packet
                       57   1E   D0  0363  4321         MOVL    #30,R7                  ; load transfer byte count
                   0000203D'EF   16  0366  4322         JSB     SEND_TU58               ; send data packet
           24 00000000'EF   00   E1  036C  4323         BBC     #V_CLOCK_FLAG,B_FLAGS,90$ ; br = no timeout error
                                     0374  4324         $DS_ERRHARD_S  ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=NO_READY
                   00001BEB'EF   DF  0374                           PUSHAL  NO_READY
                   0000190D'EF   DF  037A                           PUSHAL  T_WR_REC_ERR
                            00   DD  0380                           PUSHL   #0
                            07   DD  0382                           PUSHL   #$ER
                                     0384       ;;; TEST 10, SUBTEST 1, ERROR 7
              000100D0 9F   04   FB  0384                           CALLS   #$$M, @#DS$ERRHARD
                                     038B  4325         $DS_BNCOMPLETE  80$
                         03 50   E9  038B                           BLBC    R0, 80$
                                     038E  4326         $DS_ABORT       TEST
                            50   D4  038E                           CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  0390                           RET                     ; TERMINATE TEST
                                     0391  4327 80$:    $DS_ABORT
              00010020 9F   6C   FA  0391                           CALLG   (AP), @#DS$ABORT
                                     0398  4328                                         ; read end packet
              54    FF7E4CE0 8F   D0  0398  4329 90$:    MOVL    #DELAY_TIME7,R4         ; load read packet delay time
              55    00000000'FF   9E  039F  4330         MOVAB   @A_BUFFER_BEGIN,R5      ; load buffer address to store data
              55    000000A8 8F   C0  03A6  4331         ADDL2   #168,R5                 ; get above existing data
                  57    55   0E   C1  03AD  4332         ADDL3   #14,R5,R7               ; load last expected address of data
                   00002083'EF   16  03B1  4333         JSB     RECEIVE_TU58            ; go get data from TU58
              5A    000005F0'EF   9E  03B7  4334         MOVAB   B_END_PACKET_3,R10      ; load address of expected end packet
              5B    00000000'FF   9E  03BE  4335         MOVAB   @A_BUFFER_BEGIN,R11     ; load address of buffer
              5B    000000A8 8F   C0  03C5  4336         ADDL2   #168,R11                ; load address of actual end packet
                       57   55   D1  03CC  4337         CMPL    R5,R7                   ; check for correct number of bytes
                            31   13  03CF  4338         BEQL    130$                    ; br = correct number of bytes
              00000000'EF   08   88  03D1  4339         BISB2   #M_END_FLAG,B_FLAGS     ; set end error flag
                                     03D8  4340 95$:    $DS_ERRHARD_S  ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=REC_ERR,-
                                     03D8  4341                         P1=#T_REC_END
                   00001D2E'8F   DD  03D8                           PUSHL   #T_REC_END
                   00001EB4'EF   DF  03DE                           PUSHAL  REC_ERR
                   0000190D'EF   DF  03E4                           PUSHAL  T_WR_REC_ERR
                            00   DD  03EA                           PUSHL   #0
                            08   DD  03EC                           PUSHL   #$ER
                                     03EE       ;;; TEST 10, SUBTEST 1, ERROR 8
           000100D0 9F    05   FB  03EE                     CALLS   #$$M, @#DS$ERRHARD
                                   03F5  4342         $DS_BNCOMPLETE  100$
                   03 50   E9  03F5                     BLBC    R0, 100$
                                   03F8  4343         $DS_ABORT       TEST
                      50   D4  03F8                     CLRL    R0                     ; SEND A WARNING TO THE SUPERVISOR
                           04  03FA                     RET                            ; TERMINATE TEST
                               03FB  4344
                               03FB  4345 100$:   $DS_ABORT
           00010020 9F    6C   FA  03FB                     CALLG   (AP), @#DS$ABORT
                               0402  4346                                              ; check end packet
                   5C     0E   D0  0402  4347 130$:   MOVL    #14,R12                ; load number of bytes to check
                               0405  4348         $DS_BREAK
           00010058 9F    6E   FA  0405                     CALLG   (SP), @#DS$BREAK
                0000213B'EF   16   040C  4349         JSB     CHECK_PACKET           ; go check end packet
                               0412  4350         $DS_BNCOMPLETE  140$               ; br = no error
                   21 50   E9  0412                     BLBC    R0, 140$
                               0415  4351         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=DATA_ERR_END
                00001E9D'EF   DF   0415                     PUSHAL  DATA_ERR_END
                0000190D'EF   DF   041B                     PUSHAL  T_WR_REC_ERR
                          00   DD  0421                     PUSHL   #0
                          09   DD  0423                     PUSHL   #$ER
                               0425        ;;; TEST 10, SUBTEST 1, ERROR 9
           000100D0 9F    04   FB  0425                     CALLS   #$$M, @#DS$ERRHARD
                               042C  4352         $DS_BCOMPLETE   140$
                   07 50   E8  042C                     BLBS    R0, 140$
                               042F  4353         $DS_ABORT
           00010020 9F    6C   FA  042F                     CALLG   (AP), @#DS$ABORT
                               0436  4354 140$:
                               0436  4355
                               0436  4356
                               0436  4357
                               0436  4358         $DS_ENDSUB
                               0436        T10_S1_X:
    00010038 9F   00000024'EF   FA  0436                     CALLG   $S$, @#DS$ENDSUB
                               0441  4359
                               0441  4360         $DS_PAGE
```

```
0441                 .SBTTL  SUBTEST 10.2:   WRITE BLOCK
0441      ;+
0441      ; SUBTEST DESCRIPTION:
0441      ;
0441      ;       This subtest sends the TU58 a command packet to write one block
0441      ;       with 512 bytes of data. A continue flag should be returned between
0441      ;       sending each data packet and an end packet is then returned from the
0441      ;       TU58. Four 128-byte data packets are sent to fill one block. All byte
0441      ;       transactions between the host and TU58 are timed. If any timer
0441      ;       times out while expecting a data transfer to occur an error
0441      ;       is reported. The data packets contain alternating bytes of all
0441      ;       ones and all zeroes. The end packet is checked and if any errors
0441      ;       have occured, they are reported.
0441      ;
0441      ;
0441      ; SUBTEST STEPS:
0441      ;
0441      ;       SUBTEST WRITE BLOCK
0441      ;       :  Clear buffer
0441      ;       :  Point to write command packet
0441      ;       :  REPEAT
0441      ;       :  :  Send command packet byte at a time
0441      ;       :  :  Set timer
0441      ;       :  :  IF timer expired AND no response
0441      ;       :  :  :  THEN
0441      ;       :  :  :  Report no response error
0441      ;       :  :  :  Exit test
0441      ;       :  :  ENDIF
0441      ;       :  : UNTIL all bytes sent
0441      ;       :  ENDREPEAT
0441      ;       :  DO for count = 1 to 4
0441      ;       :  :  Wait for continue
0441      ;       :  :  Set timer
0441      ;       :  :  IF timer expired AND no response
0441      ;       :  :  :  THEN
0441      ;       :  :  :  Report no response error
0441      ;       :  :  :  Exit test
0441      ;       :  :  ENDIF
0441      ;       :  :  Point to data packet
0441      ;       :  :  REPEAT
0441      ;       :  :  :  Send data packet byte at a time
0441      ;       :  :  :  Set timer
0441      ;       :  :  :  IF timer expired AND no response
0441      ;       :  :  :  :  THEN
0441      ;       :  :  :  :  Report no response error
0441      ;       :  :  :  :  Exit test
0441      ;       :  :  :  ENDIF
0441      ;       :  :  : UNTIL all bytes sent
0441      ;       :  :  ENDREPEAT
0441      ;       :  ENDDO
0441      ;       :  REPEAT
0441      ;       :  :  Set timer
0441      ;       :  :  IF timer expired AND done set
0441      ;       :  :  :  THEN
0441      ;       :  :  :  :  Read byte
```

```
                                     0441  4417 ;       :  :  ENDIF
                                     0441  4418 ;       :  : UNTIL timer expired AND done clear
                                     0441  4419 ;       :  ENDREPEAT
                                     0441  4420 ;       :  IF expected byte count NEQ actual byte count
                                     0441  4421 ;       :  :  THEN
                                     0441  4422 ;       :  :  :  Report error
                                     0441  4423 ;       :  :  :  Exit test
                                     0441  4424 ;       :  ENDIF
                                     0441  4425 ;       :  REPEAT
                                     0441  4426 ;       :  :  Point to expected end packet
                                     0441  4427 ;       :  :  IF expected end packet data NEQ received end packet data
                                     0441  4428 ;       :  :  :  THEN
                                     0441  4429 ;       :  :  :  Report end packet data error
                                     0441  4430 ;       :  :  ENDIF
                                     0441  4431 ;       :  : UNTIL all data checked
                                     0441  4432 ;       :  ENDREPEAT
                                     0441  4433 ;       ENDSUBTEST WRITE BLOCK
                                     0441  4434 ;
                                     0441  4435 ;
                                     0441  4436 ;
                                     0441  4437 ;.page
                                     0441  4438 ;-
                                     0441  4439         $DS_BGNSUB
                                     0441       T10_S2::
     00010030 9F   00000030'EF   FA  0441                       CALLG   $S$, @#DS$BGNSUB
                                     044C  4440                                         ; clear buffers
                   00000000'EF   94  044C  4441         CLRB    B_FLAGS                 ; initialize flags
              58   00000000'FF   9E  0452  4442         MOVAB   @A_BUFFER_BEGIN,R8      ; load buffer starting address
              59   00000400 8F   D0  0459  4443         MOVL    #1024,R9                ; do two pages
                   00002134'EF   16  0460  4444         JSB     CLEAR_BUFFER            ; go clear buffers
                                     0466  4445                                         ; send write command
              54   FFF3CB00 8F   D0  0466  4446         MOVL    #DELAY_TIME3,R4         ; load send packet delay time
              55   0000006A'EF   9E  046D  4447         MOVAB   B_WRITE_2_CMD,R5        ; point to write command packet
                       57   0E   D0  0474  4448         MOVL    #14,R7                  ; load transfer byte count
                   0000203D'EF   16  0477  4449         JSB     SEND_TU58               ; send command packet
           24 00000000'EF   00   E1  047D  4450         BBC     #V_CLOCK_FLAG,B_FLAGS,20$ ; br = no timeout error
                                     0485  4451         $DS_ERRHARD_S  ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=NO_READY
                   00001BEB'EF   DF  0485                       PUSHAL  NO_READY
                   000018C8'EF   DF  048B                       PUSHAL  T_WR_BLK_ERR
                            00   DD  0491                       PUSHL   #0
                            01   DD  0493                       PUSHL   #$ER
                                     0495       ;;; TEST 10, SUBTEST 2, ERROR 1
              000100D0 9F   04   FB  0495                       CALLS   #$$M, @#DS$ERRHARD
                                     049C  4452         $DS_BNCOMPLETE  10$
                         03 50   E9  049C                       BLBC    R0, 10$
                                     049F  4453         $DS_ABORT       TEST
                            50   D4  049F                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                            04       04A1                       RET                     ; TERMINATE TEST
                                     04A2  4454 10$:    $DS_ABORT
              00010020 9F   6C   FA  04A2                       CALLG   (AP), @#DS$ABORT
                                     04A9  4455                                         ; read continue
                   00000003'EF   94  04A9  4456 20$:    CLRB    B_COUNT                 ; clear counter to send 4 data packets
              54   FFC2F700 8F   D0  04AF  4457 30$:    MOVL    #DELAY_TIME6,R4         ; load receive packet delay time
              55   00000000'FF   9E  04B6  4458         MOVAB   @A_BUFFER_BEGIN,R5      ; load buffer address to store data
                  57   55   01   C1  04BD  4459         ADDL3   #1,R5,R7                ; load last expected address of data
```

```
          00002083'EF   16  04C1  4460          JSB     RECEIVE_TU58              ; go get data from TU58
     5A   00000023'EF   9E  04C7  4461          MOVAB   B_CONTINUE,R10            ; load address of expected data
     5B   00000000'FF   9E  04CE  4462          MOVAB   @A_BUFFER_BEGIN,R11       ; load address of buffer
               57   55  D1  04D5  4463          CMPL    R5,R7                     ; check for correct number of bytes
                    2A  13  04D8  4464          BEQL    70$                       ; br = correct number of bytes
                            04DA  4465 35$:     $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=REC_ERR,-
                            04DA  4466                            P1=#T_REC_CONT
          00001D47'8F   DD  04DA                        PUSHL   #T_REC_CONT
          00001EB4'EF   DF  04E0                        PUSHAL  REC_ERR
          000018C8'EF   DF  04E6                        PUSHAL  T_WR_BLK_ERR
                    00  DD  04EC                        PUSHL   #0
                    02  DD  04EE                        PUSHL   #$ER
                            04F0        ;;; TEST 10, SUBTEST 2, ERROR 2
     000100D0 9F    05  FB  04F0                        CALLS   #$$M, @#DS$ERRHARD
                            04F7  4467          $DS_BNCOMPLETE  40$
                 03 50  E9  04F7                        BLBC    R0, 40$
                            04FA  4468          $DS_ABORT       TEST
                    50  D4  04FA                        CLRL    R0                ; SEND A WARNING TO THE SUPERVISOR
                    04  04FC                            RET                       ; TERMINATE TEST
                            04FD  4469
                            04FD  4470 40$:     $DS_ABORT
     00010020 9F    6C  FA  04FD                        CALLG   (AP), @#DS$ABORT
                            0504  4471                                            ; check continue
               5C   01  D0  0504  4472 70$:     MOVL    #1,R12                    ; load number of bytes to check
                            0507  4473          $DS_BREAK
     00010058 9F    6E  FA  0507                        CALLG   (SP), @#DS$BREAK
          0000213B'EF   16  050E  4474          JSB     CHECK_PACKET              ; go check continue
                            0514  4475          $DS_BNCOMPLETE  90$               ; br = no error
                 24 50  E9  0514                        BLBC    R0, 90$
                            0517  4476          $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=DATA_ERR_CONT
          00001E4D'EF   DF  0517                        PUSHAL  DATA_ERR_CONT
          000018C8'EF   DF  051D                        PUSHAL  T_WR_BLK_ERR
                    00  DD  0523                        PUSHL   #0
                    03  DD  0525                        PUSHL   #$ER
                            0527        ;;; TEST 10, SUBTEST 2, ERROR 3
     000100D0 9F    04  FB  0527                        CALLS   #$$M, @#DS$ERRHARD
                            052E  4477          $DS_BNCOMPLETE  80$
                 03 50  E9  052E                        BLBC    R0, 80$
                            0531  4478          $DS_ABORT       TEST
                    50  D4  0531                        CLRL    R0                ; SEND A WARNING TO THE SUPERVISOR
                    04  0533                            RET                       ; TERMINATE TEST
                            0534  4479 80$:     $DS_ABORT
     00010020 9F    6C  FA  0534                        CALLG   (AP), @#DS$ABORT
                            053B  4480                                            ; send data packet
     54   FFFB6C20 8F   D0  053B  4481 90$:     MOVL    #DELAY_TIME1,R4           ; load send packet delay time
     55   0000070A'EF   9E  0542  4482          MOVAB   B_DATA_PKT_2,R5           ; point to data packet
     57   00000084 8F   D0  0549  4483          MOVL    #132,R7                   ; load transfer byte count
          0000203D'EF   16  0550  4484          JSB     SEND_TU58                 ; send data packet
  24 00000000'EF    00  E1  0556  4485          BBC     #V_CLOCK_FLAG,B_FLAGS,110$ ; br = no timeout error
                            055E  4486          $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=NO_READY
          00001BEB'EF   DF  055E                        PUSHAL  NO_READY
          000018C8'EF   DF  0564                        PUSHAL  T_WR_BLK_ERR
                    00  DD  056A                        PUSHL   #0
                    04  DD  056C                        PUSHL   #$ER
                            056E        ;;; TEST 10, SUBTEST 2, ERROR 4
```

```
              000100D0 9F   04   FB  056E                       CALLS   #$$M, @#DS$ERRHARD
                                     0575  4487         $DS_BNCOMPLETE  100$
                         03 50   E9  0575                       BLBC    R0, 100$
                                     0578  4488         $DS_ABORT       TEST
                            50   D4  0578                       CLRL    R0               ; SEND A WARNING TO THE SUPERVISOR
                                 04  057A                       RET                      ; TERMINATE TEST
                                     057B  4489 100$:   $DS_ABORT
              00010020 9F   6C   FA  057B                       CALLG   (AP), @#DS$ABORT
                                     0582  4490
                   00000000'FF   94  0582  4491 110$:   CLRB    @A_BUFFER_BEGIN          ; clear buffer location with data
    FF1D 00000003'EF   01   03   9D  0588  4492         ACBB    #3,#1,B_COUNT,30$        ; br = not finished
                                     0592  4493                                          ; read end packet
              54   FFF3CB00 8F   D0  0592  4494 120$:   MOVL    #DELAY_TIME3,R4          ; load receive packet delay time
              55   00000000'FF   9E  0599  4495         MOVAB   @A_BUFFER_BEGIN,R5       ; load buffer address to store data
                  57   55   0E   C1  05A0  4496         ADDL3   #14,R5,R7                ; load last expected address of data
                   00002083'EF   16  05A4  4497         JSB     RECEIVE_TU58             ; go get data from TU58
              5A   000005FE'EF   9E  05AA  4498         MOVAB   B_END_PACKET_4,R10       ; load address of expected end packet
              5B   00000000'FF   9E  05B1  4499         MOVAB   @A_BUFFER_BEGIN,R11      ; load address of actual end packet
                       57   55   D1  05B8  4500         CMPL    R5,R7                    ; check for correct number of bytes
                            31   13  05BB  4501         BEQL    150$                     ; br = correct number of bytes
              00000000'EF   08   88  05BD  4502         BISB2   #M_END_FLAG,B_FLAGS      ; set end error flag
                                     05C4  4503 122$:   $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=REC_ERR,-
                                     05C4  4504                         P1=#T_REC_END
                   00001D2E'8F   DD  05C4                       PUSHL   #T_REC_END
                   00001EB4'EF   DF  05CA                       PUSHAL  REC_ERR
                   000018C8'EF   DF  05D0                       PUSHAL  T_WR_BLK_ERR
                            00   DD  05D6                       PUSHL   #0
                            05   DD  05D8                       PUSHL   #$ER
                                     05DA       ;;; TEST 10, SUBTEST 2, ERROR 5
              000100D0 9F   05   FB  05DA                       CALLS   #$$M, @#DS$ERRHARD
                                     05E1  4505         $DS_BNCOMPLETE  125$
                         03 50   E9  05E1                       BLBC    R0, 125$
                                     05E4  4506         $DS_ABORT       TEST
                            50   D4  05E4                       CLRL    R0               ; SEND A WARNING TO THE SUPERVISOR
                                 04  05E6                       RET                      ; TERMINATE TEST
                                     05E7  4507
                                     05E7  4508 125$:   $DS_ABORT
              00010020 9F   6C   FA  05E7                       CALLG   (AP), @#DS$ABORT
                                     05EE  4509                                          ; check end packet
                       5C   0E   D0  05EE  4510 150$:   MOVL    #14,R12                  ; load number of bytes to check
                                     05F1  4511         $DS_BREAK
              00010058 9F   6E   FA  05F1                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16  05F8  4512         JSB     CHECK_PACKET             ; go check end packet
                                     05FE  4513         $DS_BNCOMPLETE  170$             ; br = no error
                         24 50   E9  05FE                       BLBC    R0, 170$
                                     0601  4514         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=DATA_ERR_END
                   00001E9D'EF   DF  0601                       PUSHAL  DATA_ERR_END
                   000018C8'EF   DF  0607                       PUSHAL  T_WR_BLK_ERR
                            00   DD  060D                       PUSHL   #0
                            06   DD  060F                       PUSHL   #$ER
                                     0611       ;;; TEST 10, SUBTEST 2, ERROR 6
              000100D0 9F   04   FB  0611                       CALLS   #$$M, @#DS$ERRHARD
                                     0618  4515         $DS_BNCOMPLETE  160$
                         03 50   E9  0618                       BLBC    R0, 160$
                                     061B  4516         $DS_ABORT       TEST
                       50   D4  061B                       CLRL     R0                ; SEND A WARNING TO THE SUPERVISOR
                            04  061D                       RET                        ; TERMINATE TEST
                                061E   4517 160$:   $DS_ABORT
           00010020 9F    6C    FA  061E                       CALLG    (AP), a#DS$ABORT
                                0625   4518 170$:
                                0625   4519         $DS_ENDSUB
                                0625        T10_S2_X:
    00010038 9F   00000030'EF   FA  0625                       CALLG    $S$, a#DS$ENDSUB
                                0630   4520
                                0630   4521         $DS_PAGE
```

```
0630                  .SBTTL  SUBTEST 10.3:   READ BLOCK
0630          ;+
0630          ; SUBTEST DESCRIPTION:
0630          ;
0630          ;       This subtest sends the TU58 a command packet to read the 512
0630          ;       bytes of data that were written in the previous subtest. A
0630          ;       timer is set which will time out if there is no response from the
0630          ;       TU58 between each byte transaction. The end packet and data packets
0630          ;       are checked for errors. If any errors are encountered they are
0630          ;       reported.
0630          ;
0630          ;
0630          ; SUBTEST STEPS:
0630          ;
0630          ;       SUBTEST READ BLOCK
0630          ;       :  Clear buffer
0630          ;       :  Point to read command packet
0630          ;       :  REPEAT
0630          ;       :  :  Send command packet byte at a time
0630          ;       :  :  Set timer
0630          ;       :  :  IF timer expired and no response
0630          ;       :  :  :  THEN
0630          ;       :  :  :  Report no response error
0630          ;       :  :  :  Exit test
0630          ;       :  :  ENDIF
0630          ;       :  : UNTIL all bytes sent
0630          ;       :  ENDREPEAT
0630          ;       :  REPEAT
0630          ;       :  :  Set timer
0630          ;       :  :  IF timer expired AND done set
0630          ;       :  :  :  THEN
0630          ;       :  :  :  :  Read byte
0630          ;       :  :  ENDIF
0630          ;       :  : UNTIL timer expired AND done clear
0630          ;       :  ENDREPEAT
0630          ;       :  IF expected byte count NEQ actual byte count
0630          ;       :  :  THEN
0630          ;       :  :  :  Report error
0630          ;       :  :  :  Exit test
0630          ;       :  ENDIF
0630          ;       :  REPEAT
0630          ;       :  :  Point to expected data packet
0630          ;       :  :  REPEAT
0630          ;       :  :  :  IF data expected NEQ data received
0630          ;       :  :  :  :  THEN
0630          ;       :  :  :  :  Report data error
0630          ;       :  :  :  :  Exit subtest
0630          ;       :  :  :  ENDIF
0630          ;       :  :  :  Point to next byte of data
0630          ;       :  :  : UNTIL all data in packet checked
0630          ;       :  :  ENDREPEAT
0630          ;       :  : UNTIL 4 data packets checked
0630          ;       :  ENDREPEAT
0630          ;       :  Point to expected end packet
0630          ;       :  REPEAT
```

```
                                     0630  4578 ;       :  :  IF data expected NEQ data received
                                     0630  4579 ;       :  :  :  THEN
                                     0630  4580 ;       :  :  :  Report data error
                                     0630  4581 ;       :  :  :  Exit subtest
                                     0630  4582 ;       :  :  ENDIF
                                     0630  4583 ;       :  : UNTIL all end packet bytes checked
                                     0630  4584 ;       :  ENDREPEAT
                                     0630  4585 ;       ENDSUBTEST READ BLOCK
                                     0630  4586 ;
                                     0630  4587 ;
                                     0630  4588 ;
                                     0630  4589 ;.page
                                     0630  4590 ;-
                                     0630  4591
                                     0630  4592         $DS_BGNSUB
                                     0630       T10_S3::
     00010030 9F    0000003C'EF    FA  0630                       CALLG   $S$, @#DS$BGNSUB
                                     063B  4593                                         ; clear buffers
                   00000000'EF    94  063B  4594         CLRB    B_FLAGS                 ; initialize flags
              58    00000000'FF    9E  0641  4595         MOVAB   @A_BUFFER_BEGIN,R8      ; load buffer starting address
              59    00000400 8F    D0  0648  4596         MOVL    #1024,R9                ; do two pages
                   00002134'EF    16  064F  4597         JSB     CLEAR_BUFFER            ; go clear buffers
                                     0655  4598                                         ; send read command
              54    FFF3CB00 8F    D0  0655  4599         MOVL    #DELAY_TIME3,R4         ; load send packet delay time
              55    00000078'EF    9E  065C  4600         MOVAB   B_READ_2_CMD,R5         ; point to read command packet
                       57    0E    D0  0663  4601         MOVL    #14,R7                  ; load transfer byte count
                   0000203D'EF    16  0666  4602         JSB     SEND_TU58               ; send command packet
           24 00000000'EF    00    E1  066C  4603         BBC     #V_CLOCK_FLAG,B_FLAGS,2$ ; br = no timeout error
                                     0674  4604         $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=NO_READY
                   00001BEB'EF    DF  0674                       PUSHAL  NO_READY
                   000018EB'EF    DF  067A                       PUSHAL  T_RD_BLK_ERR
                 ,           00    DD  0680                       PUSHL   #0
                             01    DD  0682                       PUSHL   #$ER
                                     0684       ;;; TEST 10, SUBTEST 3, ERROR 1
              000100D0 9F    04    FB  0684                       CALLS   #$$M, @#DS$ERRHARD
                                     068B  4605         $DS_BNCOMPLETE  1$
                          03 50    E9  068B                       BLBC    R0, 1$
                                     068E  4606         $DS_ABORT       TEST
                             50    D4  068E                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                             04    0690                           RET                     ; TERMINATE TEST
                                     0691  4607 1$:     $DS_ABORT
              00010020 9F    6C    FA  0691                       CALLG   (AP), @#DS$ABORT
                                     0698  4608                                         ; read data and end packets
              54    FFF3CB00 8F    D0  0698  4609 2$:     MOVL    #DELAY_TIME3,R4         ; load receive packet delay time
              55    00000000'FF    9E  069F  4610         MOVAB   @A_BUFFER_BEGIN,R5      ; load buffer address to store data
         57   55    0000021E 8F    C1  06A6  4611         ADDL3   #542,R5,R7              ; load last expected address of data
                   00002083'EF    16  06AE  4612         JSB     RECEIVE_TU58            ; go get data from TU58
              5B    00000000'FF    9E  06B4  4613         MOVAB   @A_BUFFER_BEGIN,R11     ; load address of buffer
              5A    0000070A'EF    9E  06BB  4614         MOVAB   B_DATA_PKT_2,R10        ; load address of expected data
                       57    55    D1  06C2  4615         CMPL    R5,R7                   ; check for correct number of bytes
                             31    13  06C5  4616         BEQL    6$                      ; br = correct number of bytes
              00000000'EF    04    88  06C7  4617         BISB2   #M_DATA_FLAG,B_FLAGS    ; set data error flag
                                     06CE  4618 3$:     $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=REC_ERR,-
                                     06CE  4619                     P1=#T_REC_DATA
                   00001D68'8F    DD  06CE                       PUSHL   #T_REC_DATA
                   00001EB4'EF   DF  06D4                       PUSHAL  REC_ERR
                   000018EB'EF   DF  06DA                       PUSHAL  T_RD_BLK_ERR
                             00  DD  06E0                       PUSHL   #0
                             02  DD  06E2                       PUSHL   #$ER
                                     06E4       ;;; TEST 10, SUBTEST 3, ERROR 2
              000100D0 9F    05  FB  06E4                       CALLS   #$$M, @#DS$ERRHARD
                                     06EB  4620         $DS_BNCOMPLETE  4$
                         03 50   E9  06EB                       BLBC    R0, 4$
                                     06EE  4621         $DS_ABORT       TEST
                             50  D4  06EE                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  06F0                       RET                     ; TERMINATE TEST
                                     06F1  4622
                                     06F1  4623 4$:     $DS_ABORT
              00010020 9F    6C  FA  06F1                       CALLG   (AP), @#DS$ABORT
                                     06F8  4624                                         ; check data packets
                   00000006'EF   D4  06F8  4625 6$:     CLRL    L_COUNT                 ; clear counter to check 4 packets
              5C   00000084 8F   D0  06FE  4626         MOVL    #132,R12                ; load number of bytes to check
                                     0705  4627 7$:     $DS_BREAK
              00010058 9F    6E  FA  0705                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16  070C  4628         JSB     CHECK_PACKET            ; go check data packet
                                     0712  4629         $DS_BNCOMPLETE  9$              ; br = no error
                         24 50   E9  0712                       BLBC    R0, 9$
                                     0715  4630         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=DATA_ERR DATA
                   00001E75'EF   DF  0715                       PUSHAL  DATA_ERR_DATA
                   000018EB'EF   DF  071B                       PUSHAL  T_RD_BLK_ERR
                             00  DD  0721                       PUSHL   #0
                             03  DD  0723                       PUSHL   #$ER
                                     0725       ;;; TEST 10, SUBTEST 3, ERROR 3
              000100D0 9F    04  FB  0725                       CALLS   #$$M, @#DS$ERRHARD
                                     072C  4631         $DS_BNCOMPLETE  8$
                         03 50   E9  072C                       BLBC    R0, 8$
                                     072F  4632         $DS_ABORT       TEST
                             50  D4  072F                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  0731                       RET                     ; TERMINATE TEST
                                     0732  4633 8$:     $DS_ABORT
              00010020 9F    6C  FA  0732                       CALLG   (AP), @#DS$ABORT
              5B   00000084 8F   C0  0739  4634 9$:     ADDL2   #132,R11                ; adjust pointer to next packet
           BD 00000006'EF    03  F3  0740  4635         AOBLEQ  #3,L_COUNT,7$           ; br = not finished checking data
                                     0748  4636                                         ; check end packet
              5A   000005FE'EF   9E  0748  4637         MOVAB   B_END_PACKET_4,R10      ; load address of expected end packet
                        5C   0E  D0  074F  4638         MOVL    #14,R12                 ; load number of bytes to check
                                     0752  4639         $DS_BREAK
              00010058 9F    6E  FA  0752                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16  0759  4640         JSB     CHECK_PACKET            ; go check end packet
                                     075F  4641         $DS_BNCOMPLETE  11$             ; br = no error
                         24 50   E9  075F                       BLBC    R0, 11$
                                     0762  4642         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=DATA_ERR_END
                   00001E9D'EF   DF  0762                       PUSHAL  DATA_ERR_END
                   000018EB'EF   DF  0768                       PUSHAL  T_RD_BLK_ERR
                             00  DD  076E                       PUSHL   #0
                             04  DD  0770                       PUSHL   #$ER
                                     0772       ;;; TEST 10, SUBTEST 3, ERROR 4
              000100D0 9F    04  FB  0772                       CALLS   #$$M, @#DS$ERRHARD
                                     0779  4643         $DS_BCOMPLETE   10$
                         03 50   E8  0779                       BLBS    R0, 10$
```

```
                                     077C  4644          $DS_ABORT       TEST
                           50   D4  077C                          CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                04  077E                          RET                     ; TERMINATE TEST
                                     077F  4645 10$:     $DS_ABORT
             00010020 9F    6C   FA  077F                          CALLG   (AP), a#DS$ABORT
                                     0786  4646 11$:
                                     0786  4647
                                     0786  4648          $DS_ENDSUB
                                     0786       T10_S3_X:
      00010038 9F    0000003C'EF   FA  0786                          CALLG   $S$, a#DS$ENDSUB
                                     0791  4649
                                     0791  4650          $DS_PAGE
```

```
0791            .SBTTL  SUBTEST 10.4:   WRITE MODIFIED
0791   4653 ;+
0791   4654 ; SUBTEST DESCRIPTION:
0791   4655 ;
0791   4656 ;       This subtest sends the TU58 a command packet to write modified
0791   4657 ;       512 bytes of data. A timer is set which will time out if a
0791   4658 ;       response is not received within the allotted time on all byte
0791   4659 ;       transactions between the host and TU58. Four 128 byte data packets
0791   4660 ;       are sent. Before each data packet is sent a continue is read from the
0791   4661 ;       TU58 and when all data has been sent the end packet is read from the
0791   4662 ;       TU58 and checked. If any errors are encountered they are reported.
0791   4663 ;       This modified command will write all of the data and then back up and
0791   4664 ;       check the checksum for data just written.
0791   4665 ;
0791   4666 ;
0791   4667 ; SUBTEST STEPS:
0791   4668 ;
0791   4669 ;       SUBTEST WRITE MODIFIED
0791   4670 ;       :  Clear buffer
0791   4671 ;       :  Point to write command packet
0791   4672 ;       :  REPEAT
0791   4673 ;       :  :  Send command packet byte at a time
0791   4674 ;       :  :  Set timer
0791   4675 ;       :  :  IF timer expired AND no response
0791   4676 ;       :  :  :  THEN
0791   4677 ;       :  :  :  Report no response error
0791   4678 ;       :  :  :  Exit test
0791   4679 ;       :  :  ENDIF
0791   4680 ;       :  : UNTIL all bytes sent
0791   4681 ;       :  ENDREPEAT
0791   4682 ;       :  DO for count = 1 to 4
0791   4683 ;       :  :  Wait for continue
0791   4684 ;       :  :  Set timer
0791   4685 ;       :  :  IF timer expired AND no continue
0791   4686 ;       :  :  :  THEN
0791   4687 ;       :  :  :  Report no continue error
0791   4688 ;       :  :  :  Exit test
0791   4689 ;       :  :  ENDIF
0791   4690 ;       :  :  Point to data packet
0791   4691 ;       :  :  REPEAT
0791   4692 ;       :  :  :  Send data packet byte at a time
0791   4693 ;       :  :  :  Set timer
0791   4694 ;       :  :  :  IF timer expired AND no response
0791   4695 ;       :  :  :  :  THEN
0791   4696 ;       :  :  :  :  Report no response error
0791   4697 ;       :  :  :  :  Exit test
0791   4698 ;       :  :  :  ENDIF
0791   4699 ;       :  :  : UNTIL all bytes sent
0791   4700 ;       :  :  ENDREPEAT
0791   4701 ;       :  ENDDO
0791   4702 ;       :  REPEAT
0791   4703 ;       :  :  Set timer
0791   4704 ;       :  :  IF timer expired AND done set
0791   4705 ;       :  :  :  THEN
0791   4706 ;       :  :  :  :  Read byte
```

```
                                    0791  4707 ;        :  :  ENDIF
                                    0791  4708 ;        :  : UNTIL timer expired AND done clear
                                    0791  4709 ;        :  ENDREPEAT
                                    0791  4710 ;        :  IF expected byte count NEQ actual byte count
                                    0791  4711 ;        :  :  THEN
                                    0791  4712 ;        :  :  :  Report error
                                    0791  4713 ;        :  :  :  Exit test
                                    0791  4714 ;        :  ENDIF
                                    0791  4715 ;        :  Point to expected end packet
                                    0791  4716 ;        :  REPEAT
                                    0791  4717 ;        :  :  IF expected end packet data NEQ received end packet data
                                    0791  4718 ;        :  :  :  THEN
                                    0791  4719 ;        :  :  :  Report end packet data error
                                    0791  4720 ;        :  :  ENDIF
                                    0791  4721 ;        :  : UNTIL all data checked
                                    0791  4722 ;        :  ENDREPEAT
                                    0791  4723 ;       ENDSUBTEST WRITE MODIFIED
                                    0791  4724 ;
                                    0791  4725 ;
                                    0791  4726 ;.page
                                    0791  4727 ;-
                                    0791  4728          $DS_BGNSUB
                                    0791       T10_S4::
   00010030 9F   00000048'EF   FA   0791                        CALLG   $S$, @#DS$BGNSUB
                                    079C  4729                                                  ; clear buffers
                 00000000'EF   94   079C  4730          CLRB    B_FLAGS                         ; initialize flags
           58    00000000'FF   9E   07A2  4731          MOVAB   @A_BUFFER_BEGIN,R8              ; load buffer starting address
           59    00000400 8F   D0   07A9  4732          MOVL    #1024,R9                        ; do two pages
                 00002134'EF   16   07B0  4733          JSB     CLEAR_BUFFER                    ; go clear buffers
                                    07B6  4734                                                  ; send write command
           54    FFF3CB00 8F   D0   07B6  4735          MOVL    #DELAY_TIME3,R4                 ; load send packet delay time
           55    000000A2'EF   9E   07BD  4736          MOVAB   B_WRITE_4_CMD,R5                ; point to write command packet
                      57  0E   D0   07C4  4737          MOVL    #14,R7                          ; load transfer byte count
                 0000203D'EF   16   07C7  4738          JSB     SEND_TU58                       ; send command packet
         24 00000000'EF   00   E1   07CD  4739          BBC     #V_CLOCK_FLAG,B_FLAGS,2$ ; br = no timeout error
                                    07D5  4740          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_MOD_ERR,PRLINK-NO_READY
                 00001BEB'EF   DF   07D5                        PUSHAL  NO_READY
                 00001954'EF   DF   07DB                        PUSHAL  T_WR_MOD_ERR
                          00   DD   07E1                        PUSHL   #0
                          01   DD   07E3                        PUSHL   #$ER
                                    07E5       ;;; TEST 10, SUBTEST 4, ERROR 1
          000100D0 9F     04   FB   07E5                        CALLS   #$$M, @#DS$ERRHARD
                                    07EC  4741          $DS_BNCOMPLETE  1$
                       03 50   E9   07EC                        BLBC    R0, 1$
                                    07EF  4742          $DS_ABORT       TEST
                          50   D4   07EF                        CLRL    R0                      ; SEND A WARNING TO THE SUPERVISOR
                          04   07F1                             RET                             ; TERMINATE TEST
                                    07F2  4743 1$:      $DS_ABORT
          00010020 9F     6C   FA   07F2                        CALLG   (AP), @#DS$ABORT
                                    07F9  4744                                                  ; read continue
                 00000003'EF   94   07F9  4745 2$:      CLRB    B_COUNT                         ; clear counter to send 4 data packets
           54    FFF3CB00 8F   D0   07FF  4746 3$:      MOVL    #DELAY_TIME3,R4                 ; load receive packet delay time
           55    00000000'FF   9E   0806  4747          MOVAB   @A_BUFFER_BEGIN,R5              ; load buffer address to store data
              57      55  01   C1   080D  4748          ADDL3   #1,R5,R7                        ; load last expected address of data
                 00002083'EF   16   0811  4749          JSB     RECEIVE_TU58                    ; go get data from TU58
              5A   00000023'EF  9E  0817  4750         MOVAB   B_CONTINUE,R10          ; load address of expected data
              5B   00000000'FF  9E  081E  4751         MOVAB   aA_BUFFER_BEGIN,R11     ; load address of actual data
                       57   55  D1  0825  4752         CMPL    R5,R7                   ; check for correct number of byte
                            2A  13  0828  4753         BEQL    7$                      ; br = correct number of bytes
                                     082A  4754 4$:    $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_MOD_ERR,PRLINK=REC_ERR,-
                                     082A  4755                         P1=#T_REC_CONT
                   00001D47'8F  DD  082A                       PUSHL   #T_REC_CONT
                   00001EB4'EF  DF  0830                       PUSHAL  REC_ERR
                   00001954'EF  DF  0836                       PUSHAL  T_WR_MOD_ERR
                            00  DD  083C                       PUSHL   #0
                            02  DD  083E                       PUSHL   #$ER
                                     0840       ;;; TEST 10, SUBTEST 4, ERROR 2
              000100D0 9F   05  FB  0840                       CALLS   #$$M, a#DS$ERRHARD
                                     0847  4756         $DS_BNCOMPLETE  5$
                         03 50  E9  0847                       BLBC    R0, 5$
                                     084A  4757         $DS_ABORT       TEST
                            50  D4  084A                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  084C                       RET                     ; TERMINATE TEST
                                     084D  4758
                                     084D  4759 5$:    $DS_ABORT
              00010020 9F   6C  FA  084D                       CALLG   (AP), a#DS$ABORT
                                     0854  4760                                         ; check continue
                       5C   01  D0  0854  4761 7$:    MOVL    #1,R12                  ; load number of bytes to check
                                     0857  4762         $DS_BREAK
              00010058 9F   6E  FA  0857                       CALLG   (SP), a#DS$BREAK
                   0000213B'EF  16  085E  4763         JSB     CHECK_PACKET            ; go check continue
                                     0864  4764         $DS_BNCOMPLETE  9$              ; br = no error
                         24 50  E9  0864                       BLBC    R0, 9$
                                     0867  4765         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_MOD_ERR,PRLINK-DATA_ERR_CONT
                   00001E4D'EF  DF  0867                       PUSHAL  DATA_ERR_CONT
                   00001954'EF  DF  086D                       PUSHAL  T_WR_MOD_ERR
                            00  DD  0873                       PUSHL   #0
                            03  DD  0875                       PUSHL   #$ER
                                     0877       ;;; TEST 10, SUBTEST 4, ERROR 3
              000100D0 9F   04  FB  0877                       CALLS   #$$M, a#DS$ERRHARD
                                     087E  4766         $DS_BNCOMPLETE  8$
                         03 50  E9  087E                       BLBC    R0, 8$
                                     0881  4767         $DS_ABORT       TEST
                            50  D4  0881                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  0883                       RET                     ; TERMINATE TEST
                                     0884  4768 8$:    $DS_ABORT
              00010020 9F   6C  FA  0884                       CALLG   (AP), a#DS$ABORT
                                     088B  4769                                         ; send data packet
              54   FFFB6C20 8F  D0  088B  4770 9$:    MOVL    #DELAY_TIME1,R4         ; load send packet delay time
              55   0000070A'EF  9E  0892  4771         MOVAB   B_DATA_PKT_2,R5         ; point to data packet
              57   00000084 8F  D0  0899  4772         MOVL    #132,R7                 ; load transfer byte count
                   0000203D'EF  16  08A0  4773         JSB     SEND_TU58               ; send data packet
           24 00000000'EF   00  E1  08A6  4774         BBC     #V_CLOCK_FLAG,B_FLAGS,11$ ; br = no timeout error
                                     08AE  4775         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_MOD_ERR,PRLINK=NO_READY
                   00001BEB'EF  DF  08AE                       PUSHAL  NO_READY
                   00001954'EF  DF  08B4                       PUSHAL  T_WR_MOD_ERR
                            00  DD  08BA                       PUSHL   #0
                            04  DD  08BC                       PUSHL   #$ER
                                     08BE       ;;; TEST 10, SUBTEST 4, ERROR 4
              000100D0 9F   04  FB  08BE                       CALLS   #$$M, a#DS$ERRHARD
```

```
                                     08C5  4776         $DS_BNCOMPLETE  10$
                          03 50  E9  08C5                       BLBC    R0, 10$
                                     08C8  4777         $DS_ABORT       TEST
                             50  D4  08C8                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  08CA                       RET                     ; TERMINATE TEST
                                     08CB  4778 10$:    $DS_ABORT
               00010020 9F   6C  FA  08CB                       CALLG   (AP), a#DS$ABORT
                    00000000'FF  94  08D2  4779 11$:    CLRB    aA_BUFFER_BEGIN         ; clear buffer location used for data
     FF1D 00000003'EF   01   03  9D  08D8  4780         ACBB    #3,#1,B_COUNT,3$        ; br = not finished
                                     08E2  4781                                         ; read end packet
              54    FFE17B80 8F  D0  08E2  4782         MOVL    #DELAY_TIMES,R4         ; load receive packet delay time
              55    00000000'FF  9E  08E9  4783         MOVAB   aA_BUFFER_BEGIN,R5      ; load buffer address to store data
                   57   55   0E  C1  08F0  4784         ADDL3   #14,R5,R7               ; load last expected address of data
                    00002083'EF  16  08F4  4785         JSB     RECEIVE_TU58            ; go get data from TU58
              5A    000005FE'EF  9E  08FA  4786         MOVAB   B_END_PACKET_4,R10      ; load address of expected end packet
              5B    00000000'FF  9E  0901  4787         MOVAB   aA_BUFFER_BEGIN,R11     ; load address of actual end packet
                        57   55  D1  0908  4788         CMPL    R5,R7                   ; check for correct number of bytes
                             31  13  090B  4789         BEQL    15$                     ; br = correct number of bytes
              00000000'EF    08  88  090D  4790         BISB2   #M_END_FLAG,B_FLAGS     ; set end error flag
                                     0914  4791 12$:    $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_MOD_ERR,PRLINK=REC_ERR,-
                                     0914  4792                         P1=#T_REC_END
                    00001D2E'8F  DD  0914                       PUSHL   #T_REC_END
                    00001EB4'EF  DF  091A                       PUSHAL  REC_ERR
                    00001954'EF  DF  0920                       PUSHAL  T_WR_MOD_ERR
                             00  DD  0926                       PUSHL   #0
                             05  DD  0928                       PUSHL   #$ER
                                     092A       ;;; TEST 10, SUBTEST 4, ERROR 5
               000100D0 9F   05  FB  092A                       CALLS   #$$M, a#DS$ERRHARD
                                     0931  4793         $DS_BNCOMPLETE  13$
                          03 50  E9  0931                       BLBC    R0, 13$
                                     0934  4794         $DS_ABORT       TEST
                             50  D4  0934                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  0936                       RET                     ; TERMINATE TEST
                                     0937  4795
                                     0937  4796 13$:    $DS_ABORT
               00010020 9F   6C  FA  0937                       CALLG   (AP), a#DS$ABORT
                                     093E  4797                                         ; check end packet
                        5C   0E  D0  093E  4798 15$:    MOVL    #14,R12                 ; load number of bytes to check
                                     0941  4799         $DS_BREAK
               00010058 9F   6E  FA  0941                       CALLG   (SP), a#DS$BREAK
                    0000213B'EF  16  0948  4800         JSB     CHECK_PACKET            ; go check end packet
                                     094E  4801         $DS_BNCOMPLETE  17$             ; br = no error
                          24 50  E9  094E                       BLBC    R0, 17$
                                     0951  4802         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_MOD_ERR,PRLINK=DATA_ERR_END
                    00001E9D'EF  DF  0951                       PUSHAL  DATA_ERR_END
                    00001954'EF  DF  0957                       PUSHAL  T_WR_MOD_ERR
                             00  DD  095D                       PUSHL   #0
                             06  DD  095F                       PUSHL   #$ER
                                     0961       ;;; TEST 10, SUBTEST 4, ERROR 6
               000100D0 9F   04  FB  0961                       CALLS   #$$M, a#DS$ERRHARD
                                     0968  4803         $DS_BNCOMPLETE  16$
                          03 50  E9  0968                       BLBC    R0, 16$
                                     096B  4804         $DS_ABORT       TEST
                             50  D4  096B                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  096D                       RET                     ; TERMINATE TEST
```

```
                                    096E  4805 16$:     $DS_ABORT
               00010020 9F    6C    FA  096E                    CALLG   (AP), a#DS$ABORT
                                    0975  4806 17$:
                                    0975  4807          $DS_ENDSUB
                                    0975       T10_S4_X:
        00010038 9F    00000048'EF  FA  0975                    CALLG   $S$, a#DS$ENDSUB
                                    0980  4808
                                    0980  4809          $DS_PAGE
```

```
0980          .SBTTL  SUBTEST 10.5:   READ MODIFIED
0980   ;+
0980   ; SUBTEST DESCRIPTION:
0980   ;
0980   ;        This subtest sends the TU58 a command packet to read modified
0980   ;        the 512 bytes of data written in the previous subtest. This
0980   ;        command will cause the TU58 to read the tape with an
0980   ;        increased threshold. A timer is set which will time out if there
0980   ;        is no response within the allotted time. After reading the first
0980   ;        byte of data, another timer is set which will time out if an end
0980   ;        packet is not received within the allotted time. If any data or
0980   ;        timer errors are encountered, they are reported.
0980   ;
0980   ;
0980   ; SUBTEST STEPS:
0980   ;
0980   ;        SUBTEST READ MODIFIED
0980   ;        :  Clear buffer
0980   ;        :  Point to read command packet
0980   ;        :  REPEAT
0980   ;        :  :  Send command packet byte at a time
0980   ;        :  :  Set timer
0980   ;        :  :  IF timer expired and no response
0980   ;        :  :  :  THEN
0980   ;        :  :  :  Report no response error
0980   ;        :  :  :  Exit test
0980   ;        :  :  ENDIF
0980   ;        :  : UNTIL all bytes sent
0980   ;        :  ENDREPEAT
0980   ;        :  REPEAT
0980   ;        :  :  Set timer
0980   ;        :  :  IF timer expired AND done set
0980   ;        :  :  :  THEN
0980   ;        :  :  :  :  Read byte
0980   ;        :  :  ENDIF
0980   ;        :  : UNTIL timer expired AND done clear
0980   ;        :  ENDREPEAT
0980   ;        :  IF expected byte count NEQ actual byte count
0980   ;        :  :  THEN
0980   ;        :  :  :  Report error
0980   ;        :  :  :  Exit test
0980   ;        :  ENDIF
0980   ;        :  REPEAT
0980   ;        :  :  Point to expected data packet
0980   ;        :  :  REPEAT
0980   ;        :  :  :  IF data expected NEQ data received
0980   ;        :  :  :  :  THEN
0980   ;        :  :  :  :  Report data error
0980   ;        :  :  :  :  Exit subtest
0980   ;        :  :  :  ENDIF
0980   ;        :  :  :  Point to next byte of data
0980   ;        :  :  : UNTIL all data in packet checked
0980   ;        :  :  ENDREPEAT
0980   ;        :  : UNTIL 4 data packets checked
0980   ;        :  ENDREPEAT
```

```
                                     0980  4866 ;       :  Point to expected end packet
                                     0980  4867 ;       :  REPEAT
                                     0980  4868 ;       :  :  IF end packet data expected NEQ end packet data received
                                     0980  4869 ;       :  :  :  THEN
                                     0980  4870 ;       :  :  :  Report end packet data error
                                     0980  4871 ;       :  :  :  Exit subtest
                                     0980  4872 ;       :  :  ENDIF
                                     0980  4873 ;       :  : UNTIL all end packet bytes checked
                                     0980  4874 ;       :  ENDREPEAT
                                     0980  4875 ;       ENDSUBTEST READ MODIFIED
                                     0980  4876 ;
                                     0980  4877 ;
                                     0980  4878 ;
                                     0980  4879 ;.page
                                     0980  4880 ;-
                                     0980  4881         $DS_BGNSUB
                                     0980       T10_S5::
     00010030 9F   00000054'EF   FA  0980                   CALLG   $S$, @#DS$BGNSUB
                                     098B  4882                                         ; clear buffers
                   00000000'EF   94  098B  4883         CLRB    B_FLAGS                 ; initialize flags
              58   00000000'FF   9E  0991  4884         MOVAB   @A_BUFFER_BEGIN,R8      ; load buffer starting address
              59   00000400 8F   D0  0998  4885         MOVL    #1024,R9                ; do two pages
                   00002134'EF   16  099F  4886         JSB     CLEAR_BUFFER            ; go clear buffers
                                     09A5  4887                                         ; send read command
              54   FFF3CB00 8F   D0  09A5  4888         MOVL    #DELAY_TIME3,R4         ; load send packet delay time
              55   000000B0'EF   9E  09AC  4889         MOVAB   B_READ_4_CMD,R5         ; point to read command packet
                       57   0E   D0  09B3  4890         MOVL    #14,R7                  ; load transfer byte count
                   0000203D'EF   16  09B6  4891         JSB     SEND_TU58               ; send command packet
           24 00000000'EF   00   E1  09BC  4892         BBC     #V_CLOCK_FLAG,B_FLAGS,2$ ; br = no timeout error
                                     09C4  4893         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_MOD_ERR,PRLINK=NO_READY
                   00001BEB'EF   DF  09C4                   PUSHAL  NO_READY
                   0000197A'EF   DF  09CA                   PUSHAL  T_RD_MOD_ERR
                            00   DD  09D0                   PUSHL   #0
                            01   DD  09D2                   PUSHL   #$ER
                                     09D4       ;;; TEST 10, SUBTEST 5, ERROR 1
              000100D0 9F   04   FB  09D4                   CALLS   #$$M, @#DS$ERRHARD
                                     09DB  4894         $DS_BNCOMPLETE  1$
                         03 50   E9  09DB                   BLBC    R0, 1$
                                     09DE  4895         $DS_ABORT       TEST
                            50   D4  09DE                   CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  09E0                   RET                     ; TERMINATE TEST
                                     09E1  4896 1$:     $DS_ABORT
              00010020 9F   6C   FA  09E1                   CALLG   (AP), @#DS$ABORT
                                     09E8  4897                                         ; read data and end packet
              54   FFF3CB00 8F   D0  09E8  4898 2$:     MOVL    #DELAY_TIME3,R4         ; load receive packet delay time
              55   00000000'FF   9E  09EF  4899         MOVAB   @A_BUFFER_BEGIN,R5      ; load buffer address to store data
         57   55   0000021E 8F   C1  09F6  4900         ADDL3   #542,R5,R7              ; load last expected address of data
                   00002083'EF   16  09FE  4901         JSB     RECEIVE_TU58            ; go get data from TU58
              5B   00000000'FF   9E  0A04  4902         MOVAB   @A_BUFFER_BEGIN,R11     ; load address of actual data
              5A   0000070A'EF   9E  0A0B  4903         MOVAB   B_DATA_PKT_2,R10        ; load address of expected data
                       57   55   D1  0A12  4904         CMPL    R5,R7                   ; check for correct number of bytes
                            31   13  0A15  4905         BEQL    6$                      ; br = correct number of bytes
              00000000'EF   04   88  0A17  4906         BISB2   #M_DATA_FLAG,B_FLAGS    ; set data error flag
                                     0A1E  4907 3$:     $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_MOD_ERR,PRLINK=REC_ERR,-
                                     0A1E  4908                         P1=#T_REC_DATA
         00001D68'8F    DD  0A1E                       PUSHL   #T_REC_DATA
         00001EB4'EF    DF  0A24                       PUSHAL  REC_ERR
         0000197A'EF    DF  0A2A                       PUSHAL  T_RD_MOD_ERR
                  00    DD  0A30                       PUSHL   #0
                  02    DD  0A32                       PUSHL   #$ER
                            0A34          ;;; TEST 10, SUBTEST 5, ERROR 2
     000100D0 9F    05  FB  0A34                       CALLS   #$$M, @#DS$ERRHARD
                            0A3B  4909          $DS_BNCOMPLETE  4$
               03 50    E9  0A3B                       BLBC    R0, 4$
                            0A3E  4910          $DS_ABORT       TEST
                  50    D4  0A3E                       CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                        04  0A40                       RET                         ; TERMINATE TEST
                            0A41  4911
                            0A41  4912 4$:      $DS_ABORT
     00010020 9F    6C  FA  0A41                       CALLG   (AP), @#DS$ABORT
                            0A48  4913                                             ; check data packets
         00000006'EF    D4  0A48  4914 6$:      CLRL    L_COUNT                    ; clear counter to check 4 packets
     5C   00000084 8F   D0  0A4E  4915          MOVL    #132,R12                   ; load number of bytes to check
                            0A55  4916 7$:      $DS_BREAK
     00010058 9F    6E  FA  0A55                       CALLG   (SP), @#DS$BREAK
         0000213B'EF    16  0A5C  4917          JSB     CHECK_PACKET               ; go check data packet
                            0A62  4918          $DS_BNCOMPLETE  9$                 ; br = no error
               24 50    E9  0A62                       BLBC    R0, 9$
                            0A65  4919          $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_MOD_ERR,PRLINK=DATA_ERR_DATA
         00001E75'EF    DF  0A65                       PUSHAL  DATA_ERR_DATA
         0000197A'EF    DF  0A6B                       PUSHAL  T_RD_MOD_ERR
                  00    DD  0A71                       PUSHL   #0
                  03    DD  0A73                       PUSHL   #$ER
                            0A75          ;;; TEST 10, SUBTEST 5, ERROR 3
     000100D0 9F    04  FB  0A75                       CALLS   #$$M, @#DS$ERRHARD
                            0A7C  4920          $DS_BNCOMPLETE  8$
               03 50    E9  0A7C                       BLBC    R0, 8$
                            0A7F  4921          $DS_ABORT       TEST
                  50    D4  0A7F                       CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                        04  0A81                       RET                         ; TERMINATE TEST
                            0A82  4922 8$:      $DS_ABORT
     00010020 9F    6C  FA  0A82                       CALLG   (AP), @#DS$ABORT
     5B   00000084 8F   C0  0A89  4923 9$:      ADDL2   #132,R11                   ; adjust pointer to next packet
  BD 00000006'EF    03  F3  0A90  4924          AOBLEQ  #3,L_COUNT,7$              ; br = not finished checking data
                            0A98  4925                                             ; check end packet
     5A    000005FE'EF  9E  0A98  4926          MOVAB   B_END_PACKET_4,R10         ; load address of expected end packet
               5C   0E  D0  0A9F  4927          MOVL    #14,R12                    ; load number of bytes to check
                            0AA2  4928          $DS_BREAK
     00010058 9F    6E  FA  0AA2                       CALLG   (SP), @#DS$BREAK
         0000213B'EF    16  0AA9  4929          JSB     CHECK_PACKET               ; go check end packet
                            0AAF  4930          $DS_BNCOMPLETE  11$                ; br = no error
               24 50    E9  0AAF                       BLBC    R0, 11$
                            0AB2  4931          $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_MOD_ERR,PRLINK=DATA ERR_END
         00001E9D'EF    DF  0AB2                       PUSHAL  DATA_ERR_END
         0000197A'EF    DF  0AB8                       PUSHAL  T_RD_MOD_ERR
                  00    DD  0ABE                       PUSHL   #0
                  04    DD  0AC0                       PUSHL   #$ER
                            0AC2          ;;; TEST 10, SUBTEST 5, ERROR 4
     000100D0 9F    04  FB  0AC2                       CALLS   #$$M, @#DS$ERRHARD
                            0AC9  4932          $DS_BNCOMPLETE  10$
```

```
                     03 50   E9  0AC9                      BLBC    R0, 10$
                                 0ACC  4933         $DS_ABORT      TEST
                        50   D4  0ACC                      CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                             04  0ACE                      RET                         ; TERMINATE TEST
                                 0ACF  4934 10$:    $DS_ABORT
            00010020 9F   6C   FA  0ACF                      CALLG   (AP), a#DS$ABORT
                                 0AD6  4935 11$:    $DS_ENDSUB
                                 0AD6       T10_S5_X:
     00010038 9F   00000054'EF   FA  0AD6                      CALLG   $S$, a#DS$ENDSUB
                                 0AE1  4936         $DS_PAGE
```

```
0AE1                 .SBTTL  SUBTEST 10.6:   WRITE RECORD
0AE1         ;+
0AE1         ; SUBTEST DESCRIPTION:
0AE1         ;
0AE1         ;       This subtest sends the TU58 a command packet to write one record
0AE1         ;       with 128 bytes of data. A continue flag is returned and the
0AE1         ;       data packet is sent. All byte transactions between the host and
0AE1         ;       TU58 are timed and will timeout if there is no response within
0AE1         ;       the allotted time. If a timer expires while expecting a data
0AE1         ;       transfer to occur, an error is reported. The data packet
0AE1         ;       contains alternating bytes of all one's and all zero's. Both
0AE1         ;       end and data packets are checked and if any errors have occured,
0AE1         ;       they are reported.
0AE1         ;
0AE1         ;
0AE1         ; SUBTEST STEPS:
0AE1         ;
0AE1         ;       SUBTEST WRITE RECORD
0AE1         ;       :  Clear buffer
0AE1         ;       :  Point to write command packet
0AE1         ;       :  REPEAT
0AE1         ;       :  :  Send command packet byte at a time
0AE1         ;       :  :  Set timer
0AE1         ;       :  :  IF timer expired AND no response
0AE1         ;       :  :  :  THEN
0AE1         ;       :  :  :  Report no response error
0AE1         ;       :  :  :  Exit test
0AE1         ;       :  :  ENDIF
0AE1         ;       :  : UNTIL all bytes sent
0AE1         ;       :  ENDREPEAT
0AE1         ;       :  Wait for continue
0AE1         ;       :  Set timer
0AE1         ;       :  IF timer expired AND no continue
0AE1         ;       :  :  THEN
0AE1         ;       :  :  Report no continue error
0AE1         ;       :  :  Exit test
0AE1         ;       :  ENDIF
0AE1         ;       :  Point to data packet
0AE1         ;       :  REPEAT
0AE1         ;       :  :  Send data packet byte at a time
0AE1         ;       :  :  Set timer
0AE1         ;       :  :  IF timer expired AND no response
0AE1         ;       :  :  :  THEN
0AE1         ;       :  :  :  Report no response error
0AE1         ;       :  :  :  Exit test
0AE1         ;       :  :  ENDIF
0AE1         ;       :  : UNTIL all bytes sent
0AE1         ;       :  ENDREPEAT
0AE1         ;       :  REPEAT
0AE1         ;       :  :  Set timer
0AE1         ;       :  :  IF timer expired AND done set
0AE1         ;       :  :  :  THEN
0AE1         ;       :  :  :  :  Read byte
0AE1         ;       :  :  ENDIF
0AE1         ;       :  : UNTIL timer expired AND done clear
```

```
                                  0AE1   4993 ;          :  ENDREPEAT
                                  0AE1   4994 ;          :  IF expected byte count NEQ actual byte count
                                  0AE1   4995 ;          :  :  THEN
                                  0AE1   4996 ;          :  :  :  Report error
                                  0AE1   4997 ;          :  :  :  Exit test
                                  0AE1   4998 ;          :  ENDIF
                                  0AE1   4999 ;          :  REPEAT
                                  0AE1   5000 ;          :  :  Point to expected end packet
                                  0AE1   5001 ;          :  :  IF expected end packet data NEQ received end packet data
                                  0AE1   5002 ;          :  :  :  THEN
                                  0AE1   5003 ;          :  :  :  Report end packet data error
                                  0AE1   5004 ;          :  :  ENDIF
                                  0AE1   5005 ;          :  : UNTIL all data checked
                                  0AE1   5006 ;          :  ENDREPEAT
                                  0AE1   5007 ;         ENDSUBTEST WRITE RECORD
                                  0AE1   5008 ;
                                  0AE1   5009 ;
                                  0AE1   5010 ;
                                  0AE1   5011 ;.page
                                  0AE1   5012 ;-
                                  0AE1   5013
                                  0AE1   5014           $DS_BGNSUB
                                  0AE1        T10_S6::
    00010030 9F    00000060'EF    FA  0AE1                     CALLG   $S$, @#DS$BGNSUB
                   00000000'EF    94  0AEC   5015       CLRB    B_FLAGS                 ; initialize flags
                   00000002'EF    94  0AF2   5016       CLRB    B_FLAGS3
                                      0AF8   5017                                       ; clear buffers
          58       00000000'FF    9E  0AF8   5018       MOVAB   @A_BUFFER_BEGIN,R8      ; load buffer starting address
          59       00000400 8F    D0  0AFF   5019       MOVL    #1024,R9                ; do two pages
                   00002134'EF    16  0B06   5020       JSB     CLEAR_BUFFER            ; go clear buffers
                                      0B0C   5021                                       ; send write command
          54       FFE17B80 8F    D0  0B0C   5022       MOVL    #DELAY_TIME5,R4         ; load send packet delay time
          55       00000086'EF    9E  0B13   5023       MOVAB   B_WRITE_3_CMD,R5        ; point to write command packet
                         57   0E  D0  0B1A   5024       MOVL    #14,R7                  ; load transfer byte count
                   0000203D'EF    16  0B1D   5025       JSB     SEND_TU58               ; send command packet
          24 00000000'EF    00    E1  0B23   5026       BBC     #V_CLOCK_FLAG,B_FLAGS,20$ ; br = no timeout error
                                      0B2B   5027       $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=NO_READY
                   00001BEB'EF    DF  0B2B                     PUSHAL  NO_READY
                   0000190D'EF    DF  0B31                     PUSHAL  T_WR_REC_ERR
                            00    DD  0B37                     PUSHL   #0
                            01    DD  0B39                     PUSHL   #$ER
                                      0B3B        ;;; TEST 10, SUBTEST 6, ERROR 1
             000100D0 9F    04    FB  0B3B                     CALLS   #$$M, @#DS$ERRHARD
                                      0B42   5028       $DS_BNCOMPLETE  10$
                         03 50    E9  0B42                     BLBC    R0, 10$
                                      0B45   5029       $DS_ABORT        TEST
                            50    D4  0B45                     CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                            04        0B47                     RET                     ; TERMINATE TEST
                                      0B48   5030 10$:  $DS_ABORT
             00010020 9F    6C    FA  0B48                     CALLG   (AP), @#DS$ABORT
                                      0B4F   5031                                       ; read continue
          54       FFF3CB00 8F    D0  0B4F   5032 20$:  MOVL    #DELAY_TIME3,R4         ; load receive packet delay time
          55       00000000'FF    9E  0B56   5033       MOVAB   @A_BUFFER_BEGIN,R5      ; load buffer address to store data
               57   55    01      C1  0B5D   5034       ADDL3   #1,R5,R7                ; load last expected address of data
                   00002083'EF    16  0B61   5035       JSB     RECEIVE_TU58            ; go get data from TU58
```

```
              5A   00000023'EF  9E   0B67  5036         MOVAB  B_CONTINUE,R10          ; load address of expected data
              5B   00000000'FF  9E   0B6E  5037         MOVAB  @A_BUFFER_BEGIN,R11     ; load address of actual data
                       57   55  D1   0B75  5038         CMPL   R5,R7                   ; check for correct number of bytes
                            2A  13   0B78  5039         BEQL   60$                     ; br = correct number of bytes
                                     0B7A  5040 25$:    $DS_ERRHARD_S  ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=REC_ERR,-
                                     0B7A  5041                        P1=#T_REC_CONT
                   00001D47'8F  DD   0B7A                      PUSHL   #T_REC_CONT
                   00001EB4'EF  DF   0B80                      PUSHAL  REC_ERR
                   0000190D'EF  DF   0B86                      PUSHAL  T_WR_REC_ERR
                            00  DD   0B8C                      PUSHL   #0
                            02  DD   0B8E                      PUSHL   #$ER
                                     0B90       ;;; TEST 10, SUBTEST 6, ERROR 2
              000100D0 9F   05  FB   0B90                      CALLS   #$$M, @#DS$ERRHARD
                                     0B97  5042         $DS_BNCOMPLETE 30$
                         03 50  E9   0B97                      BLBC    R0, 30$
                                     0B9A  5043         $DS_ABORT      TEST
                            50  D4   0B9A                      CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                            04       0B9C                      RET                     ; TERMINATE TEST
                                     0B9D  5044
                                     0B9D  5045 30$:    $DS_ABORT
              00010020 9F   6C  FA   0B9D                      CALLG   (AP), @#DS$ABORT
                                     0BA4  5046                                        ; check continue
                       5C   01  D0   0BA4  5047 60$:    MOVL   #1,R12                  ; load number of bytes to check
                                     0BA7  5048         $DS_BREAK
              00010058 9F   6E  FA   0BA7                      CALLG   (SP), @#DS$BREAK
                   0000213B'EF  16   0BAE  5049         JSB    CHECK_PACKET            ; go check end packet
                                     0BB4  5050         $DS_BNCOMPLETE 80$             ; br = no error
                         24 50  E9   0BB4                      BLBC    R0, 80$
                                     0BB7  5051         $DS_ERRHARD_S  ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=DATA_ERR_CONT
                   00001E4D'EF  DF   0BB7                      PUSHAL  DATA_ERR_CONT
                   0000190D'EF  DF   0BBD                      PUSHAL  T_WR_REC_ERR
                            00  DD   0BC3                      PUSHL   #0
                            03  DD   0BC5                      PUSHL   #$ER
                                     0BC7       ;;; TEST 10, SUBTEST 6, ERROR 3
              000100D0 9F   04  FB   0BC7                      CALLS   #$$M, @#DS$ERRHARD
                                     0BCE  5052         $DS_BNCOMPLETE 70$
                         03 50  E9   0BCE                      BLBC    R0, 70$
                                     0BD1  5053         $DS_ABORT      TEST
                            50  D4   0BD1                      CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                            04       0BD3                      RET                     ; TERMINATE TEST
                                     0BD4  5054 70$:    $DS_ABORT
              00010020 9F   6C  FA   0BD4                      CALLG   (AP), @#DS$ABORT
                                     0BDB  5055                                        ; send data packet
              54   FFFB6C20 8F  D0   0BDB  5056 80$:    MOVL   #DELAY_TIME1,R4         ; load send packet delay time
              55   0000070A'EF  9E   0BE2  5057         MOVAB  B_DATA_PKT_2,R5         ; point to data packet
              57   00000084 8F  D0   0BE9  5058         MOVL   #132,R7                 ; load transfer byte count
                   0000203D'EF  16   0BF0  5059         JSB    SEND_TU58               ; send data packet
           24 00000000'EF   00  E1   0BF6  5060         BBC    #V_CLOCK_FLAG,B_FLAGS,100$ ; br = no timeout error
                                     0BFE  5061         $DS_ERRHARD_S  ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=NO_READY
                   00001BEB'EF  DF   0BFE                      PUSHAL  NO_READY
                   0000190D'EF  DF   0C04                      PUSHAL  T_WR_REC_ERR
                            00  DD   0C0A                      PUSHL   #0
                            04  DD   0C0C                      PUSHL   #$ER
                                     0C0E       ;;; TEST 10, SUBTEST 6, ERROR 4
              000100D0 9F   04  FB   0C0E                      CALLS   #$$M, @#DS$ERRHARD
```

```
                         0C15  5062          $DS_BNCOMPLETE  90$
           03 50    E9   0C15                        BLBC    R0, 90$
                         0C18  5063          $DS_ABORT       TEST
              50    D4   0C18                        CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                    04   0C1A                        RET                         ; TERMINATE TEST
                         0C1B  5064 90$:     $DS_ABORT
  00010020 9F    6C FA   0C1B                        CALLG   (AP), @#DS$ABORT
                         0C22  5065                                              ; read end packet
        00000002'EF  96  0C22  5066 100$:    INCB    B_FLAGS3                    ; set data done flag
54    FFC2F700 8F    D0  0C28  5067          MOVL    #DELAY_TIME6,R4             ; load receive packet delay time
55    00000000'FF    9E  0C2F  5068          MOVAB   @A_BUFFER_BEGIN,R5          ; load address to store actual data
     57   55   0E    C1  0C36  5069          ADDL3   #14,R5,R7                   ; load last expected address of data
        00002083'EF  16  0C3A  5070          JSB     RECEIVE_TU58                ; go get data from TU58
5A    0000060C'EF    9E  0C40  5071          MOVAB   B_END_PACKET_5,R10          ; load address of expected end packet
5B    00000000'FF    9E  0C47  5072          MOVAB   @A_BUFFER_BEGIN,R11         ; load address of actual end packet
          57    55   D1  0C4E  5073          CMPL    R5,R7                       ; check for correct number of bytes
                31   13  0C51  5074          BEQL    140$                        ; br = correct number of bytes
  00000000'EF    08  88  0C53  5075          BISB2   #M_END_FLAG,B_FLAGS         ; set end error flag
                         0C5A  5076 105$:    $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=REC_ERR,-
                         0C5A  5077                          P1=#T_REC_END
        00001D2E'8F  DD  0C5A                        PUSHL   #T_REC_END
        00001EB4'EF  DF  0C60                        PUSHAL  REC_ERR
        0000190D'EF  DF  0C66                        PUSHAL  T_WR_REC_ERR
                 00  DD  0C6C                        PUSHL   #0
                 05  DD  0C6E                        PUSHL   #$ER
                         0C70       ;;; TEST 10, SUBTEST 6, ERROR 5
  000100D0 9F    05  FB  0C70                        CALLS   #$$M, @#DS$ERRHARD
                         0C77  5078          $DS_BNCOMPLETE  110$
           03 50     E9  0C77                        BLBC    R0, 110$
                         0C7A  5079          $DS_ABORT       TEST
              50     D4  0C7A                        CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                     04  0C7C                        RET                         ; TERMINATE TEST
                         0C7D  5080
                         0C7D  5081 110$:    $DS_ABORT
  00010020 9F    6C  FA  0C7D                        CALLG   (AP), @#DS$ABORT
                         0C84  5082                                              ; check end packet
           5C    0E  D0  0C84  5083 140$:    MOVL    #14,R12                     ; load number of bytes to check
                         0C87  5084          $DS_BREAK
  00010058 9F    6E  FA  0C87                        CALLG   (SP), @#DS$BREAK
     0000213B'EF     16  0C8E  5085          JSB     CHECK_PACKET                ; go check end packet
                         0C94  5086          $DS_BNCOMPLETE  160$                ; br = no error
           24 50     E9  0C94                        BLBC    R0, 160$
                         0C97  5087          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=DATA_ERR_END
        00001E9D'EF  DF  0C97                        PUSHAL  DATA_ERR_END
        0000190D'EF  DF  0C9D                        PUSHAL  T_WR_REC_ERR
                 00  DD  0CA3                        PUSHL   #0
                 06  DD  0CA5                        PUSHL   #$ER
                         0CA7       ;;; TEST 10, SUBTEST 6, ERROR 6
  000100D0 9F    04  FB  0CA7                        CALLS   #$$M, @#DS$ERRHARD
                         0CAE  5088          $DS_BNCOMPLETE  150$
           03 50     E9  0CAE                        BLBC    R0, 150$
                         0CB1  5089          $DS_ABORT       TEST
              50     D4  0CB1                        CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                     04  0CB3                        RET                         ; TERMINATE TEST
                         0CB4  5090 150$:    $DS_ABORT
              00010020 9F    6C    FA  0CB4                          CALLG   (AP), @#DS$ABORT
                                       0CBB  5091 160$:
                                       0CBB  5092         $DS_ENDSUB
                                       0CBB       T10_S6_X:
       00010038 9F    00000060'EF  FA  0CBB                          CALLG   $S$, @#DS$ENDSUB
                                       0CC6  5093
                                       0CC6  5094         $DS_PAGE
```

```
 0CC6              .SBTTL  SUBTEST 10.7:   READ RECORD
 0CC6  5097 ;+
 0CC6  5098 ; SUBTEST DESCRIPTION:
 0CC6  5099 ;
 0CC6  5100 ;       This subtest sends the TU58 a command packet to read the 128
 0CC6  5101 ;       bytes of data that were written in the previous subtest. A
 0CC6  5102 ;       timer is set and times out if there is no response between all
 0CC6  5103 ;       byte transactions. The end packet and data packet are checked for
 0CC6  5104 ;       errors. If any errors are encountered they are reported.
 0CC6  5105 ;
 0CC6  5106 ;
 0CC6  5107 ; SUBTEST STEPS:
 0CC6  5108 ;
 0CC6  5109 ;       SUBTEST READ RECORD
 0CC6  5110 ;       :  Clear buffer
 0CC6  5111 ;       :  Point to read command packet
 0CC6  5112 ;       :  REPEAT
 0CC6  5113 ;       :  :  Send command packet byte at a time
 0CC6  5114 ;       :  :  Set timer
 0CC6  5115 ;       :  :  IF timer expired and no response
 0CC6  5116 ;       :  :  :  THEN
 0CC6  5117 ;       :  :  :  Report no response error
 0CC6  5118 ;       :  :  :  Exit test
 0CC6  5119 ;       :  :  ENDIF
 0CC6  5120 ;       :  : UNTIL all bytes sent
 0CC6  5121 ;       :  ENDREPEAT
 0CC6  5122 ;       :  REPEAT
 0CC6  5123 ;       :  :  Set timer
 0CC6  5124 ;       :  :  IF timer expired AND done set
 0CC6  5125 ;       :  :  :  THEN
 0CC6  5126 ;       :  :  :  :  Read byte
 0CC6  5127 ;       :  :  ENDIF
 0CC6  5128 ;       :  : UNTIL timer expired AND done clear
 0CC6  5129 ;       :  ENDREPEAT
 0CC6  5130 ;       :  IF expected byte count NEQ actual byte count
 0CC6  5131 ;       :  :  THEN
 0CC6  5132 ;       :  :  :  Report error
 0CC6  5133 ;       :  :  :  Exit test
 0CC6  5134 ;       :  ENDIF
 0CC6  5135 ;       :  Point to expected data packet
 0CC6  5136 ;       :  REPEAT
 0CC6  5137 ;       :  :  IF data expected NEQ data received
 0CC6  5138 ;       :  :  :  THEN
 0CC6  5139 ;       :  :  :  Report data error
 0CC6  5140 ;       :  :  :  Exit subtest
 0CC6  5141 ;       :  :  ENDIF
 0CC6  5142 ;       :  :  Point to next byte of data
 0CC6  5143 ;       :  : UNTIL all data in packet checked
 0CC6  5144 ;       :  ENDREPEAT
 0CC6  5145 ;       :  Point to expected end packet
 0CC6  5146 ;       :  REPEAT
 0CC6  5147 ;       :  :  IF expected end packet data NEQ received end packet data
 0CC6  5148 ;       :  :  :  THEN
 0CC6  5149 ;       :  :  :  Report end packet data error
 0CC6  5150 ;       :  :  :  Exit subtest
```

```
                                         0CC6  5151 ;          :  :  ENDIF
                                         0CC6  5152 ;          :  : UNTIL all end packet bytes checked
                                         0CC6  5153 ;          :  ENDREPEAT
                                         0CC6  5154 ;          ENDSUBTEST READ RECORD
                                         0CC6  5155 ;
                                         0CC6  5156 ;
                                         0CC6  5157 ;
                                         0CC6  5158 ;.page
                                         0CC6  5159 ;-
                                         0CC6  5160            $DS_BGNSUB
                                         0CC6           T10_S7::
       00010030 9F    0000006C'EF    FA  0CC6                          CALLG    $S$, a#DS$BGNSUB
                                         0CD1  5161                                                    ; clear buffers
                      00000000'EF    94  0CD1  5162            CLRB     B_FLAGS                         ; initialize flags
                58    00000000'FF    9E  0CD7  5163            MOVAB    aA_BUFFER_BEGIN,R8              ; load buffer starting address
                59    00000400 8F    D0  0CDE  5164            MOVL     #1024,R9                        ; do two pages
                      00002134'EF    16  0CE5  5165            JSB      CLEAR_BUFFER                    ; go clear buffers
                                         0CEB  5166                                                    ; send read command
                54    FFF3CB00 8F    D0  0CEB  5167            MOVL     #DELAY_TIME3,R4                 ; load send packet delay time
                55    00000094'EF    9E  0CF2  5168            MOVAB    B_READ_3_CMD,R5                 ; point to read command packet
                           57   0E   D0  0CF9  5169            MOVL     #14,R7                          ; load transfer byte count
                      0000203D'EF    16  0CFC  5170            JSB      SEND_TU58                       ; send command packet
           24 00000000'EF    00      E1  0D02  5171            BBC      #V_CLOCK_FLAG,B_FLAGS,2$ ; br = no timeout error
                                         0D0A  5172            $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=NO_READY
                      00001BEB'EF    DF  0D0A                          PUSHAL   NO_READY
                      00001931'EF    DF  0D10                          PUSHAL   T_RD_REC_ERR
                                00   DD  0D16                          PUSHL    #0
                                01   DD  0D18                          PUSHL    #$ER
                                         0D1A           ;;; TEST 10, SUBTEST 7, ERROR 1
              000100D0 9F       04   FB  0D1A                          CALLS    #$$M, a#DS$ERRHARD
                                         0D21  5173            $DS_BNCOMPLETE  1$
                             03 50   E9  0D21                          BLBC     R0, 1$
                                         0D24  5174            $DS_ABORT        TEST
                                50   D4  0D24                          CLRL     R0                     ; SEND A WARNING TO THE SUPERVISOR
                                04       0D26                          RET                             ; TERMINATE TEST
                                         0D27  5175 1$:        $DS_ABORT
              00010020 9F       6C   FA  0D27                          CALLG    (AP), a#DS$ABORT
                                         0D2E  5176                                                    ; read data and end packet
                54    FFF3CB00 8F    D0  0D2E  5177 2$:        MOVL     #DELAY_TIME3,R4                 ; load receive packet delay time
                55    00000000'FF    9E  0D35  5178            MOVAB    aA_BUFFER_BEGIN,R5              ; load buffer address to store data
           57   55    00000092 8F    C1  0D3C  5179            ADDL3    #146,R5,R7                      ; load last expected address of data
                      00002083'EF    16  0D44  5180            JSB      RECEIVE_TU58                    ; go get data from TU58
                5B    00000000'FF    9E  0D4A  5181            MOVAB    aA_BUFFER_BEGIN,R11             ; load address of actual data
                5A    0000070A'EF    9E  0D51  5182            MOVAB    B_DATA_PKT_2,R10                ; load address of expected data
                           57   55   D1  0D58  5183            CMPL     R5,R7                           ; check for correct number of bytes
                           31   13       0D5B  5184            BEQL     6$                              ; br = correct number of bytes
              00000000'EF       04   88  0D5D  5185            BISB2    #M_DATA_FLAG,B_FLAGS            ; set data error flag
                                         0D64  5186 3$:        $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=REC_ERR,-
                                         0D64  5187                             P1=#T_REC_DATA
                      00001D68'8F    DD  0D64                          PUSHL    #T_REC_DATA
                      00001EB4'EF    DF  0D6A                          PUSHAL   REC_ERR
                      00001931'EF    DF  0D70                          PUSHAL   T_RD_REC_ERR
                                00   DD  0D76                          PUSHL    #0
                                02   DD  0D78                          PUSHL    #$ER
                                         0D7A           ;;; TEST 10, SUBTEST 7, ERROR 2
```

```
          000100D0 9F    05   FB  0D7A                        CALLS   #$$M, @#DS$ERRHARD
                              0D81  5188            $DS_BNCOMPLETE  4$
                   03 50    E9  0D81                        BLBC    R0, 4$
                              0D84  5189            $DS_ABORT       TEST
                      50    D4  0D84                        CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                      04        0D86                        RET                         ; TERMINATE TEST
                              0D87  5190
                              0D87  5191 4$:        $DS_ABORT
          00010020 9F    6C   FA  0D87                        CALLG   (AP), @#DS$ABORT
                              0D8E  5192                                                ; check data packet
          5C    00000084 8F   D0  0D8E  5193 6$:        MOVL    #132,R12                    ; load number of bytes to check
                              0D95  5194            $DS_BREAK
          00010058 9F    6E   FA  0D95                        CALLG   (SP), @#DS$BREAK
                0000213B'EF   16  0D9C  5195            JSB     CHECK_PACKET                ; go check data packet
                              0DA2  5196            $DS_BNCOMPLETE  8$                  ; br = no error
                   24 50    E9  0DA2                        BLBC    R0, 8$
                              0DA5  5197            $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=DATA_ERR_DATA
                00001E75'EF   DF  0DA5                        PUSHAL  DATA_ERR_DATA
                00001931'EF   DF  0DAB                        PUSHAL  T_RD_REC_ERR
                         00   DD  0DB1                        PUSHL   #0
                         03   DD  0DB3                        PUSHL   #$ER
                              0DB5              ;;; TEST 10, SUBTEST 7, ERROR 3
          000100D0 9F    04   FB  0DB5                        CALLS   #$$M, @#DS$ERRHARD
                              0DBC  5198            $DS_BNCOMPLETE  7$
                   03 50    E9  0DBC                        BLBC    R0, 7$
                              0DBF  5199            $DS_ABORT       TEST
                      50    D4  0DBF                        CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                      04        0DC1                        RET                         ; TERMINATE TEST
                              0DC2  5200 7$:        $DS_ABORT
          00010020 9F    6C   FA  0DC2                        CALLG   (AP), @#DS$ABORT
                              0DC9  5201                                                ; check end packet
          5B    00000084 8F   C0  0DC9  5202 8$:        ADDL2   #132,R11                    ; adjust pointer to next packet
          5A    0000060C'EF   9E  0DD0  5203            MOVAB   B_END_PACKET_5,R10          ; load address of expected end packet
                    5C    0E   D0  0DD7  5204            MOVL    #14,R12                     ; load number of bytes to check
                              0DDA  5205            $DS_BREAK
          00010058 9F    6E   FA  0DDA                        CALLG   (SP), @#DS$BREAK
                0000213B'EF   16  0DE1  5206            JSB     CHECK_PACKET                ; go check end packet
                              0DE7  5207            $DS_BNCOMPLETE  10$                 ; br = no error
                   24 50    E9  0DE7                        BLBC    R0, 10$
                              0DEA  5208            $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=DATA_ERR_END
                00001E9D'EF   DF  0DEA                        PUSHAL  DATA_ERR_END
                00001931'EF   DF  0DF0                        PUSHAL  T_RD_REC_ERR
                         00   DD  0DF6                        PUSHL   #0
                         04   DD  0DF8                        PUSHL   #$ER
                              0DFA              ;;; TEST 10, SUBTEST 7, ERROR 4
          000100D0 9F    04   FB  0DFA                        CALLS   #$$M, @#DS$ERRHARD
                              0E01  5209            $DS_BNCOMPLETE  9$                  ; br = error, but continue
                   03 50    E9  0E01                        BLBC    R0, 9$
                              0E04  5210            $DS_ABORT       TEST
                      50    D4  0E04                        CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                      04        0E06                        RET                         ; TERMINATE TEST
                              0E07  5211 9$:        $DS_ABORT
          00010020 9F    6C   FA  0E07                        CALLG   (AP), @#DS$ABORT
                              0E0E  5212 10$:
                              0E0E  5213            $DS_ENDSUB
```

```
                                        0E0E         T10_S7_X:
     00010038 9F    0000006C'EF    FA   0E0E                         CALLG   $S$, a#DS$ENDSUB
                                        0E19   5214
                                        0E19   5215          $DS_PAGE
```

```
0E19              .SBTTL  SUBTEST 10.8:   WRITE BLOCK - PADDED ZEROS
0E19         ;+
0E19         ; SUBTEST DESCRIPTION:
0E19         ;
0E19         ;    This subtest sends the TU58 a series of command packets to check
0E19         ;    that a block written with less than 512 bytes of data, will
0E19         ;    pad the remainder of the block with data bytes of zeros. This
0E19         ;    is accomplished by writing two consecutive blocks each with
0E19         ;    512 bytes of all ones. Then a command packet is sent to
0E19         ;    write the first block with one byte of alternating ones and
0E19         ;    zeroes. This command should write the first byte with
0E19         ;    10101010 and the remaining 511 bytes with 00000000. The second
0E19         ;    block should still contain 512 bytes of all one's data. This data
0E19         ;    is read and checked. All end packets and data packets are checked as
0E19         ;    the subtest progresses. All byte transactions between host and TU58 are
0E19         ;    timed. If any errors are encountered, they are reported.
0E19         ;
0E19         ;
0E19         ; SUBTEST STEPS:
0E19         ;
0E19         ;
0E19         ;    SUBTEST WRITE BLOCK PADDED ZEROS
0E19         ;    : Clear buffer
0E19         ;    : Point to first write command packet
0E19         ;    : DO for count = 1 to 3
0E19         ;    : : REPEAT
0E19         ;    : : : Send command packet byte at a time
0E19         ;    : : : Set timer
0E19         ;    : : : IF timer expired AND no response
0E19         ;    : : : : THEN
0E19         ;    : : : : Report no response error
0E19         ;    : : : : Exit test
0E19         ;    : : : ENDIF
0E19         ;    : : : UNTIL all bytes sent
0E19         ;    : : ENDREPEAT
0E19         ;    : : DO for count = 1 to 4
0E19         ;    : : : Wait for continue
0E19         ;    : : : Set timer
0E19         ;    : : : IF timer expired AND no continue
0E19         ;    : : : : THEN
0E19         ;    : : : : Report no continue error
0E19         ;    : : : : Exit test
0E19         ;    : : : ENDIF
0E19         ;    : : : Point to data packet
0E19         ;    : : : REPEAT
0E19         ;    : : : : Send data packet byte at a time
0E19         ;    : : : : Set timer
0E19         ;    : : : : IF timer expired AND no response
0E19         ;    : : : : : THEN
0E19         ;    : : : : : Report no response error
0E19         ;    : : : : : Exit test
0E19         ;    : : : : ENDIF
0E19         ;    : : : : UNTIL all bytes sent
0E19         ;    : : : ENDREPEAT
0E19         ;    : : ENDDO
```

```
0E19  5272 ;       :  :  REPEAT
0E19  5273 ;       :  :  :  Set timer
0E19  5274 ;       :  :  :  IF timer expired AND done set
0E19  5275 ;       :  :  :  :  THEN
0E19  5276 ;       :  :  :  :  :  Read byte
0E19  5277 ;       :  :  :  ENDIF
0E19  5278 ;       :  :  : UNTIL timer expired AND done clear
0E19  5279 ;       :  :  ENDREPEAT
0E19  5280 ;       :  :  IF expected byte count NEQ actual byte count
0E19  5281 ;       :  :  :  THEN
0E19  5282 ;       :  :  :  :  Report error
0E19  5283 ;       :  :  :  :  Exit test
0E19  5284 ;       :  :  ENDIF
0E19  5285 ;       :  :  REPEAT
0E19  5286 ;       :  :  :  Point to expected end packet
0E19  5287 ;       :  :  :  IF expected end data NEQ received end data
0E19  5288 ;       :  :  :  :  THEN
0E19  5289 ;       :  :  :  :  Report end packet data error
0E19  5290 ;       :  :  :  ENDIF
0E19  5291 ;       :  :  : UNTIL all data checked
0E19  5292 ;       :  :  ENDREPEAT
0E19  5293 ;       :  :  Point to next write command
0E19  5294 ;       :  ENDDO
0E19  5295 ;       :  Clear buffer
0E19  5296 ;       :  Point to first read command packet
0E19  5297 ;       :  DO for count = 1 to 2
0E19  5298 ;       :  :  REPEAT
0E19  5299 ;       :  :  :  Send command packet byte at a time
0E19  5300 ;       :  :  :  Set timer
0E19  5301 ;       :  :  :  IF timer expired and no response
0E19  5302 ;       :  :  :  :  THEN
0E19  5303 ;       :  :  :  :  Report no response error
0E19  5304 ;       :  :  :  :  Exit test
0E19  5305 ;       :  :  :  ENDIF
0E19  5306 ;       :  :  : UNTIL all bytes sent
0E19  5307 ;       :  :  ENDREPEAT
0E19  5308 ;       :  :  :  REPEAT
0E19  5309 ;       :  :  :  Set timer
0E19  5310 ;       :  :  :  IF timer expired AND done set
0E19  5311 ;       :  :  :  :  THEN
0E19  5312 ;       :  :  :  :  :  Read byte
0E19  5313 ;       :  :  :  ENDIF
0E19  5314 ;       :  :  : UNTIL timer expired AND done clear
0E19  5315 ;       :  :  ENDREPEAT
0E19  5316 ;       :  :  IF expected byte count NEQ actual byte count
0E19  5317 ;       :  :  :  THEN
0E19  5318 ;       :  :  :  :  Report error
0E19  5319 ;       :  :  :  :  Exit test
0E19  5320 ;       :  :  ENDIF
0E19  5321 ;       :  :  REPEAT
0E19  5322 ;       :  :  :  Point to expected data packet
0E19  5323 ;       :  :  :  REPEAT
0E19  5324 ;       :  :  :  :  IF data expected NEQ data received
0E19  5325 ;       :  :  :  :  :  THEN
0E19  5326 ;       :  :  :  :  :  Report data error
```

```
                                     0E19  5327 ;       :  :  :  :  :  Exit subtest
                                     0E19  5328 ;       :  :  :  :  ENDIF
                                     0E19  5329 ;       :  :  :  :  Point to next byte of data
                                     0E19  5330 ;       :  :  :  : UNTIL all data in packet checked
                                     0E19  5331 ;       :  :  :  ENDREPEAT
                                     0E19  5332 ;       :  :  : UNTIL 4 data packets checked
                                     0E19  5333 ;       :  :  ENDREPEAT
                                     0E19  5334 ;       :  :  Point to expected end packet
                                     0E19  5335 ;       :  :  REPEAT
                                     0E19  5336 ;       :  :  :  IF end data expected NEQ end data received
                                     0E19  5337 ;       :  :  :  :  THEN
                                     0E19  5338 ;       :  :  :  :  Report end packet data error
                                     0E19  5339 ;       :  :  :  :  Exit subtest
                                     0E19  5340 ;       :  :  :  ENDIF
                                     0E19  5341 ;       :  :  : UNTIL all end packet bytes checked
                                     0E19  5342 ;       :  :  ENDREPEAT
                                     0E19  5343 ;       :  ENDDO
                                     0E19  5344 ;       ENDSUBTEST WRITE BLOCK PADDED ZEROS
                                     0E19  5345 ;
                                     0E19  5346 ;
                                     0E19  5347 ;
                                     0E19  5348 ;.page
                                     0E19  5349 ;-
                                     0E19  5350         $DS_BGNSUB
                                     0E19       T10_S8::
     00010030 9F   00000078'EF   FA  0E19                       CALLG   $S$, @#DS$BGNSUB
                   00000004'EF   94  0E24  5351         CLRB    B_COUNT1                 ; initialize command counter
              00000005'EF   03   90  0E2A  5352         MOVB    #3,B_COUNT2              ; init data packet counter
     0000000E'EF   00000084 8F   D0  0E31  5353         MOVL    #132,L_COUNT3            ; init data counter
              53   0000132B'EF   9E  0E3C  5354         MOVAB   COMMAND_TBL_1,R3         ; load pointer to command table
                                     0E43  5355                                          ; clear buffers
                   00000000'EF   94  0E43  5356 10$:    CLRB    B_FLAGS                  ; initialize flags
              58   00000000'FF   9E  0E49  5357         MOVAB   @A_BUFFER_BEGIN,R8       ; load buffer starting address
              59   00000600 8F   D0  0E50  5358         MOVL    #BUFFER_SIZE,R9          ; do all pages
                   00002134'EF   16  0E57  5359         JSB     CLEAR_BUFFER             ; go clear buffers
                                     0E5D  5360                                          ; send write command
              54   FFF3CB00 8F   D0  0E5D  5361         MOVL    #DELAY_TIME3,R4          ; load send packet delay time
                       55   93   DE  0E64  5362         MOVAL   @(R3)+,R5                ; point to write command packet
                       57   0E   D0  0E67  5363         MOVL    #14,R7                   ; load transfer byte count
                   0000203D'EF   16  0E6A  5364         JSB     SEND_TU58                ; send command packet
           24 00000000'EF   00   E1  0E70  5365         BBC     #V_CLOCK_FLAG,B_FLAGS,30$ ; br = no timeout error
                                     0E78  5366         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=NO_READY
                   00001BEB'EF   DF  0E78                       PUSHAL  NO_READY
                   000018C8'EF   DF  0E7E                       PUSHAL  T_WR_BLK_ERR
                            00   DD  0E84                       PUSHL   #0
                            01   DD  0E86                       PUSHL   #$ER
                                     0E88       ;;; TEST 10, SUBTEST 8, ERROR 1
              000100D0 9F   04   FB  0E88                       CALLS   #$$M, @#DS$ERRHARD
                                     0E8F  5367         $DS_BNCOMPLETE  20$
                         03 50   E9  0E8F                       BLBC    R0, 20$
                                     0E92  5368         $DS_ABORT       TEST
                            50   D4  0E92                       CLRL    R0               ; SEND A WARNING TO THE SUPERVISOR
                                 04  0E94                       RET                      ; TERMINATE TEST
                                     0E95  5369 20$:    $DS_ABORT
              00010020 9F   6C   FA  0E95                       CALLG   (AP), @#DS$ABORT
```

```
                                  0E9C  5370                                                ; read continue
           00000003'EF      94    0E9C  5371 30$:      CLRB     B_COUNT                     ; initialize data packet counter
      54   FFC2F700 8F      D0    0EA2  5372 35$:      MOVL     #DELAY_TIME6,R4             ; load receive packet delay time
      55   00000000'FF      9E    0EA9  5373           MOVAB    @A_BUFFER_BEGIN,R5          ; load buffer address to store data
          57   55   01      C1    0EB0  5374           ADDL3    #1,R5,R7                    ; load last expected address of data
           00002083'EF      16    0EB4  5375           JSB      RECEIVE_TU58                ; go get data from TU58
      5A   00000023'EF      9E    0EBA  5376           MOVAB    B_CONTINUE,R10              ; load address of expected data
      5B   00000000'FF      9E    0EC1  5377           MOVAB    @A_BUFFER_BEGIN,R11         ; load address of actual data
                57   55     D1    0EC8  5378           CMPL     R5,R7                       ; check for correct number of bytes
                     2A     13    0ECB  5379           BEQL     70$                         ; br = correct number of bytes
                                  0ECD  5380 37$:      $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=REC_ERR,
                                  0ECD  5381                            P1=#T_REC_CONT
           00001D47'8F      DD    0ECD                          PUSHL    #T_REC_CONT
           00001EB4'EF      DF    0ED3                          PUSHAL   REC_ERR
           000018C8'EF      DF    0ED9                          PUSHAL   T_WR_BLK_ERR
                     00     DD    0EDF                          PUSHL    #0
                     02     DD    0EE1                          PUSHL    #$ER
                                  0EE3             ;;; TEST 10, SUBTEST 8, ERROR 2
      000100D0 9F    05     FB    0EE3                          CALLS    #$$M, @#DS$ERRHARD
                                  0EEA  5382           $DS_BNCOMPLETE   40$
                  03 50     E9    0EEA                          BLBC     R0, 40$
                                  0EED  5383           $DS_ABORT        TEST
                     50     D4    0EED                          CLRL     R0                  ; SEND A WARNING TO THE SUPERVISOR
                            04    0EEF                          RET                          ; TERMINATE TEST
                                  0EF0  5384
                                  0EF0  5385 40$:      $DS_ABORT
      00010020 9F    6C     FA    0EF0                          CALLG    (AP), @#DS$ABORT
                                  0EF7  5386                                                ; check continue
                5C   01     D0    0EF7  5387 70$:      MOVL     #1,R12                      ; load number of bytes to check
                                  0EFA  5388           $DS_BREAK
      00010058 9F    6E     FA    0EFA                          CALLG    (SP), @#DS$BREAK
           0000213B'EF      16    0F01  5389           JSB      CHECK_PACKET                ; go check continue
                                  0F07  5390           $DS BNCOMPLETE   90$                 ; br = no error
                  24 50     E9    0F07                          BLBC     R0, 90$
                                  0F0A  5391           $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=DATA_ERR_CONT
           00001E4D'EF      DF    0F0A                          PUSHAL   DATA_ERR_CONT
           000018C8'EF      DF    0F10                          PUSHAL   T_WR_BLK_ERR
                     00     DD    0F16                          PUSHL    #0
                     03     DD    0F18                          PUSHL    #$ER
                                  0F1A             ;;; TEST 10, SUBTEST 8, ERROR 3
      000100D0 9F    04     FB    0F1A                          CALLS    #$$M, @#DS$ERRHARD
                                  0F21  5392           $DS_BNCOMPLETE   80$
                  03 50     E9    0F21                          BLBC     R0, 80$
                                  0F24  5393           $DS_ABORT        TEST
                     50     D4    0F24                          CLRL     R0                  ; SEND A WARNING TO THE SUPERVISOR
                            04    0F26                          RET                          ; TERMINATE TEST
                                  0F27  5394 80$:      $DS_ABORT
      00010020 9F    6C     FA    0F27                          CALLG    (AP), @#DS$ABORT
                                  0F2E  5395                                                ; send data packet
      54   FFFB6C20 8F      D0    0F2E  5396 90$:      MOVL     #DELAY_TIME1,R4             ; load send packet delay time
                55   93     DE    0F35  5397           MOVAL    @(R3)+,R5                   ; point to data packet
      57   0000000E'EF      D0    0F38  5398           MOVL     L_COUNT3,R7                 ; load transfer byte count
           0000203D'EF      16    0F3F  5399           JSB      SEND_TU58                   ; send data packet
   24 00000000'EF    00     E1    0F45  5400           BBC      #V_CLOCK_FLAG,B_FLAGS,110$  ; br = no timeout error
                                  0F4D  5401           $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=NO_READY
```

```
                   00001BEB'EF   DF  0F4D                       PUSHAL  NO_READY
                   000018C8'EF   DF  0F53                       PUSHAL  T_WR_BLK_ERR
                            00   DD  0F59                       PUSHL   #0
                            04   DD  0F5B                       PUSHL   #$ER
                                     0F5D       ;;; TEST 10, SUBTEST 8, ERROR 4
              000100D0 9F   04   FB  0F5D                       CALLS   #$$M, @#DS$ERRHARD
                                     0F64  5402         $DS_BNCOMPLETE  100$
                         03 50   E9  0F64                       BLBC    R0, 100$
                                     0F67  5403         $DS_ABORT       TEST
                            50   D4  0F67                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  0F69                       RET                     ; TERMINATE TEST
                                     0F6A  5404 100$:   $DS_ABORT
              00010020 9F   6C   FA  0F6A                       CALLG   (AP), @#DS$ABORT
                                     0F71  5405
                   00000000'FF   94  0F71  5406 110$:   CLRB    @A_BUFFER_BEGIN         ; clear buffer location used for data
00000003'EF   01   00000005'EF   9D  0F77  5407         ACBB    B_COUNT2,#1,B_COUNT,35$ ; br = not finished sending data
                               FF1D  0F83
                                     0F85  5408                                         ; read end packet
              54   FFE17B80 8F   D0  0F85  5409 115$:   MOVL    #DELAY_TIMES,R4         ; load receive packet delay time
              55   00000000'FF   9E  0F8C  5410         MOVAB   @A_BUFFER_BEGIN,R5      ; load buffe- address to store data
                  57   55   0E   C1  0F93  5411         ADDL3   #14,R5,R7               ; load last expected address of data
                   00002083'EF   16  0F97  5412         JSB     RECEIVE_TU58            ; go get data from TU58
                       5A   93   DE  0F9D  5413         MOVAL   @(R3)+,R10              ; load address of expected end packet
              5B   00000000'FF   9E  0FA0  5414         MOVAB   @A_BUFFER_BEGIN,R11     ; load address of actual end packet
                       57   55   D1  0FA7  5415         CMPL    R5,R7                   ; check for correct number of bytes
                            31   13  0FAA  5416         BEQL    150$                    ; br = correct number of bytes
              00000000'EF   08   88  0FAC  5417         BISB2   #M_END_FLAG,B_FLAGS     ; set end error flag
                                     0FB3  5418 117$:   $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=REC_ERR,-
                                     0FB3  5419                         P1=#T_REC_END
                   00001D2E'8F   DD  0FB3                       PUSHL   #T_REC_END
                   00001EB4'EF   DF  0FB9                       PUSHAL  REC_ERR
                   000018C8'EF   DF  0FBF                       PUSHAL  T_WR_BLK_ERR
                            00   DD  0FC5                       PUSHL   #0
                            05   DD  0FC7                       PUSHL   #$ER
                                     0FC9       ;;; TEST 10, SUBTEST 8, ERROR 5
              000100D0 9F   05   FB  0FC9                       CALLS   #$$M, @#DS$ERRHARD
                                     0FD0  5420         $DS_BNCOMPLETE  120$
                         03 50   E9  0FD0                       BLBC    R0, 120$
                                     0FD3  5421         $DS_ABORT       TEST
                            50   D4  0FD3                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  0FD5                       RET                     ; TERMINATE TEST
                                     0FD6  5422
                                     0FD6  5423 120$:   $DS_ABORT
              00010020 9F   6C   FA  0FD6                       CALLG   (AP), @#DS$ABORT
                                     0FDD  5424                                         ; check end packet
                       5C   0E   D0  0FDD  5425 150$:   MOVL    #14,R12                 ; load number of bytes to check
                                     0FE0  5426         $DS_BREAK
              00010058 9F   6E   FA  0FE0                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16  0FE7  5427         JSB     CHECK_PACKET            ; go check end packet
                                     0FED  5428         $DS_BNCOMPLETE  160$            ; br = no error
                         24 50   E9  0FED                       BLBC    R0, 160$
                                     0FF0  5429         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK-DATA ERR_END
                   00001E9D'EF   DF  0FF0                       PUSHAL  DATA_ERR_END
                   000018C8'EF   DF  0FF6                       PUSHAL  T_WR_BLK_ERR
                            00   DD  0FFC                       PUSHL   #0
                            06   DD  0FFE                        PUSHL   #$ER
                                     1000       ;;; TEST 10, SUBTEST 8, ERROR 6
              000100D0 9F   04   FB  1000                        CALLS   #$$M, @#DS$ERRHARD
                                     1007  5430          $DS_BNCOMPLETE  155$
                         03 50   E9  1007                        BLBC    R0, 155$
                                     100A  5431          $DS_ABORT       TEST
                            50   D4  100A                        CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  100C                        RET                     ; TERMINATE TEST
                                     100D  5432 155$:    $DS_ABORT
              00010020 9F   6C   FA  100D                        CALLG   (AP), @#DS$ABORT
                1A 00000005'EF   E9  1014  5433 160$:    BLBC    B_COUNT2,165$           ; go do read commands if writes done
    FE1E 00000004'EF   01   01   9D  101B  5434          ACBB    #1,#1,B_COUNT1,10$      ; br = not finished
                   00000005'EF   94  1025  5435          CLRB    B_COUNT2                ; set data packet counter for 1 packet
              0000000E'EF   05   D0  102B  5436          MOVL    #5,L_COUNT3             ; set data count to 5 bytes
                          FE0E   31  1032  5437          BRW     10$                     ; go do last write
                   00000004'EF   94  1035  5438 165$:    CLRB    B_COUNT1                ; initialize command counter
                                     103B  5439                                          ; clear buffers
                   00000000'EF   94  103B  5440 170$:    CLRB    B_FLAGS                 ; initialize flags
              58   00000000'FF   9E  1041  5441          MOVAB   @A_BUFFER_BEGIN,R8      ; load buffer starting address
              59   00000600 8F   D0  1048  5442          MOVL    #BUFFER_SIZE,R9         ; do all pages
                   00002134'EF   16  104F  5443          JSB     CLEAR_BUFFER            ; go clear buffers
                                     1055  5444                                          ; send read command
              54   FFFB6C20 8F   D0  1055  5445          MOVL    #DELAY_TIME1,R4         ; load send packet delay time
                       55   93   DE  105C  5446          MOVAL   @(R3)+,R5               ; point to read command packet
                       57   0E   D0  105F  5447          MOVL    #14,R7                  ; load transfer byte count
                   0000203D'EF   16  1062  5448          JSB     SEND_TU58               ; send command packet
           24 00000000'EF   00   E1  1068  5449          BBC     #V_CLOCK_FLAG,B_FLAGS,190$ ; br = no timeout error
                                     1070  5450          $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=NO_READY
                   00001BEB'EF   DF  1070                        PUSHAL  NO_READY
                   000018EB'EF   DF  1076                        PUSHAL  T_RD_BLK_ERR
                            00   DD  107C                        PUSHL   #0
                            07   DD  107E                        PUSHL   #$ER
                                     1080       ;;; TEST 10, SUBTEST 8, ERROR 7
              000100D0 9F   04   FB  1080                        CALLS   #$$M, @#DS$ERRHARD
                                     1087  5451          $DS_BNCOMPLETE  180$
                         03 50   E9  1087                        BLBC    R0, 180$
                                     108A  5452          $DS_ABORT       TEST
                            50   D4  108A                        CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  108C                        RET                     ; TERMINATE TEST
                                     108D  5453 180$:    $DS_ABORT
              00010020 9F   6C   FA  108D                        CALLG   (AP), @#DS$ABORT
                                     1094  5454                                          ; read data and end packets
              54   FFE17B80 8F   D0  1094  5455 190$:    MOVL    #DELAY_TIME5,R4         ; load receive packet delay time
              55   00000000'FF   9E  109B  5456          MOVAB   @A_BUFFER_BEGIN,R5      ; load buffer address to store data
         57   55   0000021E 8F   C1  10A2  5457          ADDL3   #542,R5,R7              ; load last expected address of data
                   00002083'EF   16  10AA  5458          JSB     RECEIVE_TU58            ; go get data from TU58
              5B   00000000'FF   9E  10B0  5459          MOVAB   @A_BUFFER_BEGIN,R11     ; load address of actual data
                       5A   93   DE  10B7  5460          MOVAL   @(R3)+,R10              ; load address of expected data
                       57   55   D1  10BA  5461          CMPL    R5,R7                   ; check for correct number of bytes
                            31   13  10BD  5462          BEQL    230$                    ; br = correct number of bytes
              00000000'EF   04   88  10BF  5463          BISB2   #M_DATA_FLAG,B_FLAGS    ; set data error flag
                                     10C6  5464 195$:    $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=REC_ERR,-
                                     10C6  5465                                  P1=#T_REC_DATA
                   00001D68'8F   DD  10C6                        PUSHL   #T_REC_DATA
                   00001EB4'EF   DF  10CC                        PUSHAL  REC_ERR
                   000018EB'EF   DF   10D2                       PUSHAL  T_RD_BLK_ERR
                            00   DD   10D8                       PUSHL   #0
                            08   DD   10DA                       PUSHL   #$ER
                                      10DC       ;;; TEST 10, SUBTEST 8, ERROR 8
              000100D0 9F   05   FB   10DC                       CALLS   #$$M, @#DS$ERRHARD
                                      10E3  5466         $DS_BNCOMPLETE  200$
                         03 50   E9   10E3                       BLBC    R0, 200$
                                      10E4  5467         $DS_ABORT       TEST
                            50   D4   10E6                       CLRL    R0                ; SEND A WARNING TO THE SUPERVISOR
                                 04   10E8                       RET                       ; TERMINATE TEST
                                      10E9  5468
                                      10E9  5469 200$:   $DS_ABORT
              00010020 9F   6C   FA   10E9                       CALLG   (AP), @#DS$ABORT
                                      10F0  5470                                           ; check data packets
                   00000003'EF   94   10F0  5471 230$:   CLRB    B_COUNT                   ; clear counter to check 4 packets
              5C   00000084 8F   D0   10F6  5472         MOVL    #132,R12                  ; load number of bytes to check
                                      10FD  5473 240$:   $DS_BREAK
              00010058 9F   6E   FA   10FD                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16   1104  5474         JSB     CHECK_PACKET              ; go check data packet
                                      110A  5475         $DS_BNCOMPLETE  260$              ; br = no error
                         24 50   E9   110A                       BLBC    R0, 260$
                                      110D  5476         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=DATA_ERR_DATA
                   00001E75'EF   DF   110D                       PUSHAL  DATA_ERR_DATA
                   000018EB'EF   DF   1113                       PUSHAL  T_RD_BLK_ERR
                            00   DD   1119                       PUSHL   #0
                            09   DD   111B                       PUSHL   #$ER
                                      111D       ;;; TEST 10, SUBTEST 8, ERROR 9
              000100D0 9F   04   FB   111D                       CALLS   #$$M, @#DS$ERRHARD
                                      1124  5477         $DS_BNCOMPLETE  250$
                         03 50   E9   1124                       BLBC    R0, 250$
                                      1127  5478         $DS_ABORT       TEST
                            50   D4   1127                       CLRL    R0                ; SEND A WARNING TO THE SUPERVISOR
                                 04   1129                       RET                       ; TERMINATE TEST
                                      112A  5479 250$:   $DS_ABORT
              00010020 9F   6C   FA   112A                       CALLG   (AP), @#DS$ABORT
              5B   00000084 8F   C0   1131  5480 260$:   ADDL2   #132,R11                  ; adjust pointer to next packet
                       5A   93   DE   1138  5481         MOVAL   @(R3)+,R10                ; load address of expected data
    FFB8 00000003'EF   01   03   9D   113B  5482         ACBB    #3,#1,B_COUNT,240$        ; br = not finished checking data
                                      1145  5483                                           ; check end packet
                       5C   0E   D0   1145  5484         MOVL    #14,R12                   ; load number of bytes to check
                                      1148  5485         $DS_BREAK
              00010058 9F   6E   FA   1148                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16   114F  5486         JSB     CHECK_PACKET              ; go check end packet
                                      1155  5487         $DS_BNCOMPLETE  270$              ; br = no error
                         24 50   E9   1155                       BLBC    R0, 270$
                                      1158  5488         $DS_ERRHARD_S   ,UNIT-#0,MSGADR=T_RD_BLK_ERR,PRLINK=DATA_ERR_END
                   00001E9D'EF   DF   1158                       PUSHAL  DATA_ERR_END
                   000018EB'EF   DF   115E                       PUSHAL  T_RD_BLK_ERR
                            00   DD   1164                       PUSHL   #0
                            0A   DD   1166                       PUSHL   #$ER
                                      1168       ;;; TEST 10, SUBTEST 8, ERROR 10
              000100D0 9F   04   FB   1168                       CALLS   #$$M, @#DS$ERRHARD
                                      116F  5489         $DS_BNCOMPLETE  265$
                         03 50   E9   116F                       BLBC    R0, 265$
                                      1172  5490         $DS_ABORT       TEST
```

```
                        50   D4  1172                       CLRL    R0                   ; SEND A WARNING TO THE SUPERVISOR
                             04  1174                       RET                          ; TERMINATE TEST
                                 1175   5491 265$:   $DS_ABORT
           00010020 9F    6C   FA  1175                       CALLG   (AP), @#DS$ABORT
    FEB5 00000003'EF   01   01   9D  117C   5492 270$:   ACBB    #1,#1,B_COUNT,170$       ; br = not finished reading
                                 1186   5493
                                 1186   5494
                                 1186   5495
                                 1186   5496
                                 1186   5497         $DS_ENDSUB
                                 1186        T10_S8_X:
     00010038 9F    00000078'EF   FA  1186                       CALLG   $S$, @#DS$ENDSUB
                                 1191   5498
                                 1191   5499         $DS_PAGE
```

```
1191                 .SBTTL  SUBTEST 10.9:   WRITE RECORD - PADDED ZEROS
1191       ;+
1191       ; SUBTEST DESCRIPTION:
1191       ;
1191       ;         This subtest sends the TU58 a series of command packets to check
1191       ;         that a record written with less than 128 bytes of data, will
1191       ;         pad the remainder of the record with data bytes of zeroes. This
1191       ;         is accomplished by writing two consecutive records each with
1191       ;         128 bytes of all ones. Then a command packet is sent to
1191       ;         write the first record with one byte of alternating ones and
1191       ;         zeroes data. This command should write the first byte with
1191       ;         10101010 and the remaining 127 bytes with 00000000. The second
1191       ;         block should still contain 128 bytes of all ones data. This data
1191       ;         is read and checked. All end packets and data packets are checked as
1191       ;         the subtest progresses. All byte transactions between the host and TU58
1191       ;         are timed. If any errors are encountered, they are reported.
1191       ;
1191       ;
1191       ; SUBTEST STEPS:
1191       ;
1191       ;
1191       ;         SUBTEST WRITE RECORD PADDED ZEROS
1191       ;         :  Clear buffer
1191       ;         :  Point to first write command packet
1191       ;         :  DO for count = 1 to 3
1191       ;         :  :  REPEAT
1191       ;         :  :  :  Send command packet byte at a time
1191       ;         :  :  :  Set timer
1191       ;         :  :  :  IF timer expired AND no response
1191       ;         :  :  :  :  THEN
1191       ;         :  :  :  :  Report no response error
1191       ;         :  :  :  :  Exit test
1191       ;         :  :  :  ENDIF
1191       ;         :  :  : UNTIL all bytes sent
1191       ;         :  :  ENDREPEAT
1191       ;         :  :  Wait for continue
1191       ;         :  :  Set timer
1191       ;         :  :  IF timer expired AND no continue
1191       ;         :  :  :  THEN
1191       ;         :  :  :  Report no continue error
1191       ;         :  :  :  Exit test
1191       ;         :  :  ENDIF
1191       ;         :  :  Point to data packet
1191       ;         :  :  REPEAT
1191       ;         :  :  :  Send data packet byte at a time
1191       ;         :  :  :  Set timer
1191       ;         :  :  :  IF timer expired AND no response
1191       ;         :  :  :  :  THEN
1191       ;         :  :  :  :  Report no response error
1191       ;         :  :  :  :  Exit test
1191       ;         :  :  :  ENDIF
1191       ;         :  :  : UNTIL all bytes sent
1191       ;         :  :  ENDREPEAT
1191       ;         :  :  REPEAT
1191       ;         :  :  :  Set timer
```

```
1191  5556 ;          :  :  :  IF timer expired AND done set
1191  5557 ;          :  :  :  :  THEN
1191  5558 ;          :  :  :  :  :  Read byte
1191  5559 ;          :  :  :  ENDIF
1191  5560 ;          :  :  : UNTIL timer expired AND done clear
1191  5561 ;          :  :  ENDREPEAT
1191  5562 ;          :  :  IF expected byte count NEQ actual byte count
1191  5563 ;          :  :  :  THEN
1191  5564 ;          :  :  :  :  Report error
1191  5565 ;          :  :  :  :  Exit test
1191  5566 ;          :  :  ENDIF
1191  5567 ;          :  :  REPEAT
1191  5568 ;          :  :  :  Point to expected end packet
1191  5569 ;          :  :  :  IF expected end data NEQ received end data
1191  5570 ;          :  :  :  :  THEN
1191  5571 ;          :  :  :  :  Report end packet data error
1191  5572 ;          :  :  :  ENDIF
1191  5573 ;          :  :  : UNTIL all data checked
1191  5574 ;          :  :  ENDREPEAT
1191  5575 ;          :  :  Point to next write command
1191  5576 ;          :  ENDDO
1191  5577 ;          :  Clear buffer
1191  5578 ;          :  Point to first read command packet
1191  5579 ;          :  DO for count = 1 to 2
1191  5580 ;          :  :  REPEAT
1191  5581 ;          :  :  :  Send command packet byte at a time
1191  5582 ;          :  :  :  Set timer
1191  5583 ;          :  :  :  IF timer expired and no response
1191  5584 ;          :  :  :  :  THEN
1191  5585 ;          :  :  :  :  Report no response error
1191  5586 ;          :  :  :  :  Exit test
1191  5587 ;          :  :  :  ENDIF
1191  5588 ;          :  :  : UNTIL all bytes sent
1191  5589 ;          :  :  ENDREPEAT
1191  5590 ;          :  :  REPEAT
1191  5591 ;          :  :  :  Set timer
1191  5592 ;          :  :  :  IF timer expired AND done set
1191  5593 ;          :  :  :  :  THEN
1191  5594 ;          :  :  :  :  :  Read byte
1191  5595 ;          :  :  :  ENDIF
1191  5596 ;          :  :  : UNTIL timer expired AND done clear
1191  5597 ;          :  :  ENDREPEAT
1191  5598 ;          :  :  IF expected byte count NEQ actual byte count
1191  5599 ;          :  :  :  THEN
1191  5600 ;          :  :  :  :  Report error
1191  5601 ;          :  :  :  :  Exit test
1191  5602 ;          :  :  ENDIF
1191  5603 ;          :  :  Point to expected data packet
1191  5604 ;          :  :  REPEAT
1191  5605 ;          :  :  :  IF data expected NEQ data received
1191  5606 ;          :  :  :  :  THEN
1191  5607 ;          :  :  :  :  Report data error
1191  5608 ;          :  :  :  :  Exit subtest
1191  5609 ;          :  :  :  ENDIF
1191  5610 ;          :  :  :  Point to next byte of data
```

```
                                    1191  5611 ;          :  :  : UNTIL all data in packet checked
                                    1191  5612 ;          :  :  ENDREPEAT
                                    1191  5613 ;          :  :  Point to expected end packet
                                    1191  5614 ;          :  :  REPEAT
                                    1191  5615 ;          :  :  :  IF end data expected NEQ end data received
                                    1191  5616 ;          :  :  :  :  THEN
                                    1191  5617 ;          :  :  :  :  Report end packet data error
                                    1191  5618 ;          :  :  :  :  Exit subtest
                                    1191  5619 ;          :  :  :  ENDIF
                                    1191  5620 ;          :  :  : UNTIL all end packet bytes checked
                                    1191  5621 ;          :  :  ENDREPEAT
                                    1191  5622 ;          :  ENDDO
                                    1191  5623 ;          ENDSUBTEST WRITE RECORD PADDED ZEROS
                                    1191  5624 ;
                                    1191  5625 ;
                                    1191  5626 ;
                                    1191  5627 ;.page
                                    1191  5628 ;-
                                    1191  5629             $DS_BGNSUB
                                    1191       T10_S9::
    00010030 9F   00000084'EF   FA  1191                        CALLG   $S$, @#DS$BGNSUB
                                    119C  5630
                  00000004'EF   94  119C  5631             CLRB     B_COUNT1                ; initialize command counter
    0000000E'EF   00000084 8F   D0  11A2  5632             MOVL     #132,L_COUNT3           ; init byte count
             53   00001397'EF   9E  11AD  5633             MOVAB    COMMAND_TBL_2,R3        ; load pointer to command table
                                    11B4  5634                                              ; clear buffers
                  00000000'EF   94  11B4  5635 10$:        CLRB     B_FLAGS                 ; initialize flags
             58   00000000'FF   9E  11BA  5636             MOVAB    @A_BUFFER_BEGIN,R8      ; load buffer starting address
             59   00000600 8F   D0  11C1  5637             MOVL     #BUFFER_SIZE,R9         ; do all pages
                  00002134'EF   16  11C8  5638             JSB      CLEAR_BUFFER            ; go clear buffers
                                    11CE  5639                                              ; send write command
             54   FFF3CB00 8F   D0  11CE  5640             MOVL     #DELAY_TIME3,R4         ; load send packet delay time
                        55   93 DE  11D5  5641             MOVAL    @(R3)+,R5               ; point to write command packet
                        57   0E D0  11D8  5642             MOVL     #14,R7                  ; load transfer byte count
                  0000203D'EF   16  11DB  5643             JSB      SEND_TU58               ; send command packet
           24 00000000'EF    00 E1  11E1  5644             BBC      #V_CLOCK_FLAG,B_FLAGS,30$ ; br = no timeout error
                                    11E9  5645             $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=NO_READY
                  00001BEB'EF   DF  11E9                        PUSHAL  NO_READY
                  0000190D'EF   DF  11EF                        PUSHAL  T_WR_REC_ERR
                             00 DD  11F5                        PUSHL   #0
                             01 DD  11F7                        PUSHL   #$ER
                                    11F9       ::: TEST 10, SUBTEST 9, ERROR 1
          000100D0 9F        04 FB  11F9                        CALLS   #$$M, @#DS$ERRHARD
                                    1200  5646             $DS_BNCOMPLETE  20$
                          03 50 E9  1200                        BLBC    R0, 20$
                                    1203  5647             $DS_ABORT        TEST
                             50 D4  1203                        CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                04  1205                        RET                     ; TERMINATE TEST
                                    1206  5648 20$:        $DS_ABORT
          00010020 9F        6C FA  1206                        CALLG   (AP), @#DS$ABORT
                                    120D  5649                                              ; read continue
          54   FFF3CB00 8F      D0  120D  5650 30$:        MOVL     #DELAY_TIME3,R4         ; load receive packet delay time
          55   00000000'FF      9E  1214  5651             MOVAB    @A_BUFFER_BEGIN,R5      ; load buffer address to store data
               57   55   01     C1  121B  5652             ADDL3    #1,R5,R7                ; load last expected address of data
                  00002083'EF   16  121F  5653             JSB      RECEIVE_TU58            ; go get data from TU58
          5A   00000023'EF    9E  1225  5654          MOVAB   B_CONTINUE,R10          ; load address of expected data
          5B   00000000'FF    9E  122C  5655          MOVAB   @A_BUFFER_BEGIN,R11     ; load address of actual data
                    57   55   D1  1233  5656          CMPL    R5,R7                   ; check for correct number of bytes
                         2A   13  1236  5657          BEQL    70$                     ; br = correct number of bytes
                              1238  5658 35$:         $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=REC_ERR,-
                              1238  5659                               P1=#T_REC_CONT
               00001D47'8F    DD  1238                        PUSHL   #T_REC_CONT
               00001EB4'EF    DF  123E                        PUSHAL  REC_ERR
               0000190D'EF    DF  1244                        PUSHAL  T_WR_REC_ERR
                         00   DD  124A                        PUSHL   #0
                         02   DD  124C                        PUSHL   #$ER
                              124E          ;;; TEST 10, SUBTEST 9, ERROR 2
        000100D0 9F      05   FB  124E                        CALLS   #$$M, @#DS$ERRHARD
                              1255  5660          $DS_BNCOMPLETE  40$
                 03 50        E9  1255                        BLBC    R0, 40$
                              1258  5661          $DS_ABORT       TEST
                         50   D4  1258                        CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                              04  125A                        RET                         ; TERMINATE TEST
                              125B  5662
                              125B  5663 40$:     $DS_ABORT
        00010020 9F      6C   FA  125B                        CALLG   (AP), @#DS$ABORT
                              1262  5664                                                  ; check continue
              5C         01   D0  1262  5665 70$:     MOVL    #1,R12                      ; load number of bytes to check
                              1265  5666          $DS_BREAK
        00010058 9F      6E   FA  1265                        CALLG   (SP), @#DS$BREAK
               0000213B'EF    16  126C  5667          JSB     CHECK_PACKET                ; go check continue
                              1272  5668          $DS_BNCOMPLETE  90$                     ; br = no error
                 24 50        E9  1272                        BLBC    R0, 90$
                              1275  5669          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=DATA_ERR_CONT
               00001E4D'EF    DF  1275                        PUSHAL  DATA_ERR_CONT
               0000190D'EF    DF  127B                        PUSHAL  T_WR_REC_ERR
                         00   DD  1281                        PUSHL   #0
                         03   DD  1283                        PUSHL   #$ER
                              1285          ;;; TEST 10, SUBTEST 9, ERROR 3
        000100D0 9F      04   FB  1285                        CALLS   #$$M, @#DS$ERRHARD
                              128C  5670          $DS_BNCOMPLETE  80$
                 03 50        E9  128C                        BLBC    R0, 80$
                              128F  5671          $DS_ABORT       TEST
                         50   D4  128F                        CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                              04  1291                        RET                         ; TERMINATE TEST
                              1292  5672 80$:     $DS_ABORT
        00010020 9F      6C   FA  1292                        CALLG   (AP), @#DS$ABORT
                              1299  5673                                                  ; send data packet
          54   FFFB6C20 8F    D0  1299  5674 90$:     MOVL    #DELAY_TIME1,R4             ; load serd packet delay time
                    55   93   DE  12A0  5675          MOVAL   @(R3)+,R5                   ; point to data packet
          57   0000000E'EF    D0  12A3  5676          MOVL    L_COUNT3,R7                 ; load transfer byte count
               0000203D'EF    16  12AA  5677          JSB     SEND_TUSB                   ; send data packet
       24 00000000'EF    00   E1  12B0  5678          BBC     #V_CLOCK_FLAG,B_FLAGS,110$ ; br = no timeout error
                              12B8  5679          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=NO_READY
               00001BEB'EF    DF  12B8                        PUSHAL  NO_READY
               0000190D'EF    DF  12BE                        PUSHAL  T_WR_REC_ERR
                         00   DD  12C4                        PUSHL   #0
                         04   DD  12C6                        PUSHL   #$ER
                              12C8          ;;; TEST 10, SUBTEST 9, ERROR 4
        000100D0 9F      04   FB  12C8                        CALLS   #$$M, @#DS$ERRHARD
```

```
                                12CF  5680         $DS_BNCOMPLETE  100$
               03 50      E9    12CF                       BLBC     R0, 100$
                                12D2  5681         $DS_ABORT         TEST
                  50      D4    12D2                       CLRL     R0                   ; SEND A WARNING TO THE SUPERVISOR
                          04    12D4                       RET                           ; TERMINATE TEST
                                12D5  5682 100$:   $DS_ABORT
      00010020 9F    6C   FA    12D5                       CALLG    (AP), @#DS$ABORT
                                12DC  5683
         00000000'FF      94    12DC  5684 110$:   CLRB     @A_BUFFER_BEGIN              ; clear buffer used to store continue
                                12E2  5685                                               ; read end packet
      54    FFC2F700 8F   D0    12E2  5686         MOVL     #DELAY_TIME6,R4              ; load receive packet delay time
      55    00000000'FF   9E    12E9  5687         MOVAB    @A_BUFFER_BEGIN,R5           ; load buffer address to store data
         57    55    0E   C1    12F0  5688         ADDL3    #14,R5,R7                    ; load last expected address of data
         00002083'EF      16    12F4  5689         JSB      RECEIVE_TU58                 ; go get data from TU58
              5A     93   DE    12FA  5690         MOVAL    @(R3)+,R10                   ; load address of expected end packet
      5B    00000000'FF   9E    12FD  5691         MOVAB    @A_BUFFER_BEGIN,R11          ; load address of actual end packet
              57     55   D1    1304  5692         CMPL     R5,R7                        ; check for correct number of bytes
                     31   13    1307  5693         BEQL     150$                         ; br = correct number of bytes
      00000000'EF    08   88    1309  5694         BISB2    #M_END_FLAG,B_FLAGS          ; set end error flag
                                1310  5695 115$:   $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=REC_ERR,-
                                1310  5696                          P1=#T_REC_END
         00001D2E'8F      DD    1310                       PUSHL    #T_REC_END
         00001EB4'EF      DF    1316                       PUSHAL   REC_ERR
         0000190D'EF      DF    131C                       PUSHAL   T_WR_REC_ERR
                  00      DD    1322                       PUSHL    #0
                  05      DD    1324                       PUSHL    #$ER
                                1326       ;;; TEST 10, SUBTEST 9, ERROR 5
      000100D0 9F    05   FB    1326                       CALLS    #$$M, @#DS$ERRHARD
                                132D  5697         $DS_BNCOMPLETE  120$
               03 50      E9    132D                       BLBC     R0, 120$
                                1330  5698         $DS_ABORT         TEST
                  50      D4    1330                       CLRL     R0                   ; SEND A WARNING TO THE SUPERVISOR
                          04    1332                       RET                           ; TERMINATE TEST
                                1333  5699
                                1333  5700 120$:   $DS_ABORT
      00010020 9F    6C   FA    1333                       CALLG    (AP), @#DS$ABORT
                                133A  5701                                               ; check end packet
              5C     0E   D0    133A  5702 150$:   MOVL     #14,R12                      ; load number of bytes to check
                                133D  5703         $DS_BREAK
      00010058 9F    6E   FA    133D                       CALLG    (SP), @#DS$BREAK
         0000213B'EF      16    1344  5704         JSB      CHECK_PACKET                 ; go check end packet
                                134A  5705         $DS_BNCOMPLETE  160$                  ; br = no error
               24 50      E9    134A                       BLBC     R0, 160$
                                134D  5706         $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=DATA_ERR_END
         00001E9D'EF      DF    134D                       PUSHAL   DATA_ERR_END
         0000190D'EF      DF    1353                       PUSHAL   T_WR_REC_ERR
                  00      DD    1359                       PUSHL    #0
                  06      DD    135B                       PUSHL    #$ER
                                135D       ;;; TEST 10, SUBTEST 9, ERROR 6
      000100D0 9F    04   FB    135D                       CALLS    #$$M, @#DS$ERRHARD
                                1364  5707         $DS_BNCOMPLETE  155$
               03 50      E9    1364                       BLBC     R0, 155$
                                1367  5708         $DS_ABORT         TEST
                  50      D4    1367                       CLRL     R0                   ; SEND A WARNING TO THE SUPERVISOR
                          04    1369                       RET                           ; TERMINATE TEST
```

```
                                   136A  5709 155$:    $DS_ABORT
          00010020 9F    6C    FA  136A                        CALLG   (AP), a#DS$ABORT
            14 0000000E'EF     E8  1371  5710 160$:    BLBS    L_COUNT3,165$             ; go do reads if writes done
     FE32 00000004'EF    01    01  9D  1378  5711       ACBB    #1,#1,B_COUNT1,10$        ; br = not finished
          0000000E'EF    05    D0  1382  5712          MOVL    #5,L_COUNT3               ; load byte count for last write
                       FE28    31  1389  5713          BRW     10$                       ; go do last write
              00000004'EF      94  138C  5714 165$:    CLRB    B_COUNT1                  ; initialize command counter
                                   1392  5715                                            ; clear buffers
              00000000'EF      94  1392  5716 170$:    CLRB    B_FLAGS                   ; initialize flags
          58    00000000'FF    9E  1398  5717          MOVAB   aA_BUFFER_BEGIN,R8        ; load buffer starting address
          59    00000600 8F    D0  139F  5718          MOVL    #BUFFER_SIZE,R9           ; do all pages
              00002134'EF      16  13A6  5719          JSB     CLEAR_BUFFER              ; go clear buffers
                                   13AC  5720                                            ; send read command
          54    FFFB6C20 8F    D0  13AC  5721          MOVL    #DELAY_TIME1,R4           ; load send packet delay time
                     55  93    DE  13B3  5722          MOVAL   a(R3)+,R5                 ; point to read command packet
                     57  0E    D0  13B6  5723          MOVL    #14,R7                    ; load transfer byte count
              0000203D'EF      16  13B9  5724          JSB     SEND_TU58                 ; send command packet
        24 00000000'EF   00    E1  13BF  5725          BBC     #V_CLOCK_FLAG,B_FLAGS,190$ ; br = no timeout error
                                   13C7  5726          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=NO_READY
              00001BEB'EF      DF  13C7                        PUSHAL  NO_READY
              00001931'EF      DF  13CD                        PUSHAL  T_RD_REC_ERR
                         00    DD  13D3                        PUSHL   #0
                         07    DD  13D5                        PUSHL   #$ER
                                   13D7          ;;; TEST 10, SUBTEST 9, ERROR 7
          000100D0 9F    04    FB  13D7                        CALLS   #$$M, a#DS$ERRHARD
                                   13DE  5727          $DS_BNCOMPLETE  180$
                    03 50      E9  13DE                        BLBC    R0, 180$
                                   13E1  5728          $DS_ABORT       TEST
                         50    D4  13E1                        CLRL    R0                    ; SEND A WARNING TO THE SUPERVISOR
                         04        13E3                        RET                           ; TERMINATE TEST
                                   13E4  5729 180$:    $DS_ABORT
          00010020 9F    6C    FA  13E4                        CALLG   (AP), a#DS$ABORT
                                   13EB  5730                                            ; read data and end packets
          54    FFF3CB00 8F    D0  13EB  5731 190$:    MOVL    #DELAY_TIME3,R4           ; load receive packet delay time
          55    00000000'FF    9E  13F2  5732          MOVAB   aA_BUFFER_BEGIN,R5        ; load buffer address to store data
     57   55    00000092 8F    C1  13F9  5733          ADDL3   #146,R5,R7                ; load last expected address of data
              00002083'EF      16  1401  5734          JSB     RECEIVE_TU58              ; go get data from TU58
          5B    00000000'FF    9E  1407  5735          MOVAB   aA_BUFFER_BEGIN,R11       ; load address of actual data
                     5A  93    DE  140E  5736          MOVAL   a(R3)+,R10                ; load address of expected data
                     57  55    D1  1411  5737          CMPL    R5,R7                     ; check for correct number of bytes
                     31  13    13  1414  5738          BEQL    230$                      ; br = correct number of bytes
          00000000'EF    04    88  1416  5739          BISB2   #M_DATA_FLAG,B_FLAGS      ; set data error flag
                                   141D  5740 195$:    $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=REC_ERR,-
                                   141D  5741                           P1=#T_REC_DATA
              00001D68'8F      DD  141D                        PUSHL   #T_REC_DATA
              00001EB4'EF      DF  1423                        PUSHAL  REC_ERR
              00001931'EF      DF  1429                        PUSHAL  T_RD_REC_ERR
                         00    DD  142F                        PUSHL   #0
                         08    DD  1431                        PUSHL   #$ER
                                   1433          ;;; TEST 10, SUBTEST 9, ERROR 8
          000100D0 9F    05    FB  1433                        CALLS   #$$M, a#DS$ERRHARD
                                   143A  5742          $DS_BNCOMPLETE  200$
                    03 50      E9  143A                        BLBC    R0, 200$
                                   143D  5743          $DS_ABORT       TEST
                         50    D4  143D                        CLRL    R0                    ; SEND A WARNING TO THE SUPERVISOR
```

```
                                 04  143F                       RET                      ; TERMINATE TEST
                                     1440  5744
                                     1440  5745 200$:   $DS_ABORT
              00010020 9F   6C   FA  1440                       CALLG   (AP), @#DS$ABORT
                                     1447  5746                                          ; check data packet
              5C   00000084 8F   D0  1447  5747 230$:   MOVL    #132,R12                 ; load number of bytes to check
                                     144E  5748 240$:   $DS_BREAK
              00010058 9F   6E   FA  144E                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16  1455  5749         JSB     CHECK_PACKET             ; go check data packet
                                     145B  5750         $DS_BNCOMPLETE  260$             ; br = no error
                         24 50   E9  145B                       BLBC    R0, 260$
                                     145E  5751         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=DATA_ERR_DATA
                   00001E75'EF   DF  145E                       PUSHAL  DATA_ERR_DATA
                   00001931'EF   DF  1464                       PUSHAL  T_RD_REC_ERR
                            00   DD  146A                       PUSHL   #0
                            09   DD  146C                       PUSHL   #$ER
                                     146E       ;;; TEST 10, SUBTEST 9, ERROR 9
              000100D0 9F   04   FB  146E                       CALLS   #$$M, @#DS$ERRHARD
                                     1475  5752         $DS_BNCOMPLETE  250$
                         03 50   E9  1475                       BLBC    R0, 250$
                                     1478  5753         $DS_ABORT       TEST
                            50   D4  1478                       CLRL    R0                       ; SEND A WARNING TO THE SUPERVISOR
                            04   04  147A                       RET                              ; TERMINATE TEST
                                     147B  5754 250$:   $DS_ABORT
              00010020 9F   6C   FA  147B                       CALLG   (AP), @#DS$ABORT
                                     1482  5755                                          ; check end packet
              5B   00000084 8F   C0  1482  5756 260$:   ADDL2   #132,R11                 ; adjust pointer to next packet
                         5A 93   DE  1489  5757         MOVAL   @(R3)+,R10               ; load address of expected end packet
                         5C 0E   D0  148C  5758         MOVL    #14,R12                  ; load number of bytes to check
                                     148F  5759         $DS_BREAK
              00010058 9F   6E   FA  148F                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16  1496  5760         JSB     CHECK_PACKET             ; go check end packet
                                     149C  5761         $DS_BNCOMPLETE  270$             ; br = no error
                         24 50   E9  149C                       BLBC    R0, 270$
                                     149F  5762         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=DATA_ERR_END
                   00001E9D'EF   DF  149F                       PUSHAL  DATA_ERR_END
                   00001931'EF   DF  14A5                       PUSHAL  T_RD_REC_ERR
                            00   DD  14AB                       PUSHL   #0
                            0A   DD  14AD                       PUSHL   #$ER
                                     14AF       ;;; TEST 10, SUBTEST 9, ERROR 10
              000100D0 9F   04   FB  14AF                       CALLS   #$$M, @#DS$ERRHARD
                                     14B6  5763         $DS_BNCOMPLETE  265$
                         03 50   E9  14B6                       BLBC    R0, 265$
                                     14B9  5764         $DS_ABORT       TEST
                            50   D4  14B9                       CLRL    R0                       ; SEND A WARNING TO THE SUPERVISOR
                            04       14BB                       RET                              ; TERMINATE TEST
                                     14BC  5765 265$:   $DS_ABORT
              00010020 9F   6C   FA  14BC                       CALLG   (AP), @#DS$ABORT
    FEC5 00000004'EF   01   01   9D  14C3  5766 270$:   ACBB    #1,#1,B_COUNT1,170$      ; br = not finished reading
                                     14CD  5767
                                     14CD  5768
                                     14CD  5769
                                     14CD  5770
                                     14CD  5771         $DS_ENDSUB
                                     14CD       T10_S9_X:
```

```
     00010038 9F   00000084'EF   FA  14CD                   CALLG    $S$, a#DS$ENDSUB
                                     14D8  5772
                                     14D8  5773        $DS_PAGE
```

```
14D8             .SBTTL  SUBTEST 10.10:   BLOCK WRITE - READ
14D8        ;+
14D8        ; SUBTEST DESCRIPTION:
14D8        ;
14D8        ;       This subtest sends the TU58 controller a series of command packets
14D8        ;       that will cause head switches, forward seeks, reverse seeks and
14D8        ;       seeks with head switches during read and write operations on
14D8        ;       blocks. All data written is read and checked. All end packets
14D8        ;       are checked as the subtest progresses. If any errors occur they are
14D8        ;       reported.
14D8        ;
14D8        ;
14D8        ; SUBTEST STEPS:
14D8        ;
14D8        ;       SUBTEST BLOCK WRITE - READ
14D8        ;       :  DO for count = 1 to 5
14D8        ;       :  :  Point to write command packet
14D8        ;       :  :  Clear buffer
14D8        ;       :  :  REPEAT
14D8        ;       :  :  :  Send command packet byte at a time
14D8        ;       :  :  :  Set timer
14D8        ;       :  :  :  IF timer expired AND no response
14D8        ;       :  :  :  :  THEN
14D8        ;       :  :  :  :  Report no response error
14D8        ;       :  :  :  :  Exit test
14D8        ;       :  :  :  ENDIF
14D8        ;       :  :  : UNTIL all bytes sent
14D8        ;       :  :  ENDREPEAT
14D8        ;       :  :  DO for count = 1 to 4
14D8        ;       :  :  :  Wait for continue
14D8        ;       :  :  :  Set timer
14D8        ;       :  :  :  IF timer expired AND no continue
14D8        ;       :  :  :  :  THEN
14D8        ;       :  :  :  :  Report no continue error
14D8        ;       :  :  :  :  Exit test
14D8        ;       :  :  :  ENDIF
14D8        ;       :  :  :  Point to data packet
14D8        ;       :  :  :  REPEAT
14D8        ;       :  :  :  :  Send data packet byte at a time
14D8        ;       :  :  :  :  Set timer
14D8        ;       :  :  :  :  IF timer expired AND no response
14D8        ;       :  :  :  :  :  THEN
14D8        ;       :  :  :  :  :  Report no response error
14D8        ;       :  :  :  :  :  Exit test
14D8        ;       :  :  :  :  ENDIF
14D8        ;       :  :  :  : UNTIL all bytes sent
14D8        ;       :  :  :  ENDREPEAT
14D8        ;       :  :  ENDDO
14D8        ;       :  :  REPEAT
14D8        ;       :  :  :  Set timer
14D8        ;       :  :  :  IF timer expired AND done set
14D8        ;       :  :  :  :  THEN
14D8        ;       :  :  :  :  :  Read byte
14D8        ;       :  :  :  ENDIF
14D8        ;       :  :  : UNTIL timer expired AND done clear
```

```
14D8   ;        :   :   ENDREPEAT
14D8   ;        :   :   IF expected byte count NEQ actual byte count
14D8   ;        :   :   :   THEN
14D8   ;        :   :   :   :   Report error
14D8   ;        :   :   :   :   Exit test
14D8   ;        :   :   ENDIF
14D8   ;        :   :   REPEAT
14D8   ;        :   :   :   Point to expected end packet
14D8   ;        :   :   :   IF expected end data NEQ received end data
14D8   ;        :   :   :   :   THEN
14D8   ;        :   :   :   :   Report end packet data error
14D8   ;        :   :   :   ENDIF
14D8   ;        :   :   : UNTIL all data checked
14D8   ;        :   :   ENDREPEAT
14D8   ;        :   :   Point to read command packet
14D8   ;        :   :   REPEAT
14D8   ;        :   :   :   Send command packet byte at a time
14D8   ;        :   :   :   Set timer
14D8   ;        :   :   :   IF timer expired and no response
14D8   ;        :   :   :   :   THEN
14D8   ;        :   :   :   :   Report no response error
14D8   ;        :   :   :   :   Exit test
14D8   ;        :   :   :   ENDIF
14D8   ;        :   :   : UNTIL all bytes sent
14D8   ;        :   :   ENDREPEAT
14D8   ;        :   :   REPEAT
14D8   ;        :   :   :   Set timer
14D8   ;        :   :   :   IF timer expired AND done set
14D8   ;        :   :   :   :   THEN
14D8   ;        :   :   :   :   :   Read byte
14D8   ;        :   :   :   ENDIF
14D8   ;        :   :   : UNTIL timer expired AND done clear
14D8   ;        :   :   ENDREPEAT
14D8   ;        :   :   IF expected byte count NEQ actual byte count
14D8   ;        :   :   :   THEN
14D8   ;        :   :   :   :   Report error
14D8   ;        :   :   :   :   Exit test
14D8   ;        :   :   ENDIF
14D8   ;        :   :   REPEAT
14D8   ;        :   :   :   Point to expected data packet
14C8   ;        :   :   :   REPEAT
14D8   ;        :   :   :   :   IF data expected NEQ data received
14D8   ;        :   :   :   :   :   THEN
14D8   ;        :   :   :   :   :   Report data error
14D8   ;        :   :   :   :   :   Exit subtest
14D8   ;        :   :   :   :   ENDIF
14D8   ;        :   :   :   :   Point to next byte of data
14D8   ;        :   :   :   : UNTIL all data in packet checked
14D8   ;        :   :   :   ENDREPEAT
14D8   ;        :   :   : UNTIL 4 data packets checked
14D8   ;        :   :   ENDREPEAT
14D8   ;        :   :   Point to expected end packet
14D8   ;        :   :   REPEAT
14D8   ;        :   :   :   IF data expected NEQ data received
14D8   ;        :   :   :   :   THEN
```

```
                                     14D8  5885 ;       :  :  :  :  Report data error
                                     14D8  5886 ;       :  :  :  :  Exit subtest
                                     14D8  5887 ;       :  :  :  ENDIF
                                     14D8  5888 ;       :  :  : UNTIL all end packet bytes checked
                                     14D8  5889 ;       :  :  ENDREPEAT
                                     14D8  5890 ;       :  ENDDO
                                     14D8  5891 ;       ENDSUBTEST BLOCK WRITE - READ
                                     14D8  5892 ;
                                     14D8  5893 ;
                                     14D8  5894 ;.page
                                     14D8  5895 ;-
                                     14D8  5896         $DS_BGNSUB
                                     14D8       T10_S10::
     00010030 9F   00000090'EF   FA  14D8                   CALLG   $S$, @#DS$BGNSUB
                   00000001'EF   94  14E3  5897         CLRB    B_FLAGS2                 ; initialize repeat flag
              00000001'EF   06   88  14E9  5898         BISB2   #<M_REPEAT_SEND+M_REPEAT_CHECK>,B_FLAGS2
                   00000004'EF   94  14F0  5899         CLRB    B_COUNT1                 ; initialize command counter
              53   000013D3'EF   9E  14F6  5900         MOVAB   COMMAND_TBL_3,R3         ; load pointer to command table
                                     14FD  5901                                          ; clear buffers
                   00000000'EF   94  14FD  5902 10$:    CLRB    B_FLAGS                  ; initialize flags
              58   00000000'FF   9E  1503  5903         MOVAB   @A_BUFFER_BEGIN,R8       ; load buffer starting address
              59   00000600 8F   D0  150A  5904         MOVL    #BUFFER_SIZE,R9          ; do all pages
                   00002134'EF   16  1511  5905         JSB     CLEAR_BUFFER             ; go clear buffers
                                     1517  5906                                          ; send write command
              54   FFF3CB00 8F   D0  1517  5907         MOVL    #DELAY_TIME3,R4          ; load send packet delay time
                       55   93   DE  151E  5908         MOVAL   @(R3)+,R5                ; point to write command packet
                       57   0E   D0  1521  5909         MOVL    #14,R7                   ; load transfer byte count
                   0000203D'EF   16  1524  5910         JSB     SEND_TU58                ; send command packet
           24 00000000'EF   00   E1  152A  5911         BBC     #V_CLOCK_FLAG,B_FLAGS,30$ ; br = no timeout error
                                     1532  5912         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=NO_READY
                   00001BEB'EF   DF  1532                   PUSHAL  NO_READY
                   000018C8'EF   DF  1538                   PUSHAL  T_WR_BLK_ERR
                            00   DD  153E                   PUSHL   #0
                            01   DD  1540                   PUSHL   #$ER
                                     1542       ;;; TEST 10, SUBTEST 10, ERROR 1
              000100D0 9F   04   FB  1542                   CALLS   #$$M, @#DS$ERRHARD
                                     1549  5913         $DS_BNCOMPLETE  20$
                         03 50   E9  1549                   BLBC    R0, 20$
                                     154C  5914         $DS_ABORT       TEST
                            50   D4  154C                   CLRL    R0               ; SEND A WARNING TO THE SUPERVISOR
                                 04  154E                   RET                      ; TERMINATE TEST
                                     154F  5915 20$:    $DS_ABORT
              00010020 9F   6C   FA  154F                   CALLG   (AP), @#DS$ABORT
                                     1556  5916                                          ; read continue
                   00000003'EF   94  1556  5917 30$:    CLRB    B_COUNT                  ; initialize data packet counter
              54   FF7E4CE0 8F   D0  155C  5918 35$:    MOVL    #DELAY_TIME7,R4          ; load receive packet delay time
              55   00000000'FF   9E  1563  5919         MOVAB   @A_BUFFER_BEGIN,R5       ; load buffer address to store data
                  57   55   01   C1  156A  5920         ADDL3   #1,R5,R7                 ; load last expected address of data
                   00002083'EF   16  156E  5921         JSB     RECEIVE_TU58             ; go get data from TU58
              5A   00000023'EF   9E  1574  5922         MOVAB   B_CONTINUE,R10           ; load address of expected continue
              5B   00000000'FF   9E  157B  5923         MOVAB   @A_BUFFER_BEGIN,R11      ; load address of buffer
                       57   55   D1  1582  5924         CMPL    R5,R7                    ; check for correct number of bytes
                            2A   13  1585  5925         BEQL    70$                      ; br = correct number of bytes
                                     1587  5926 37$:    $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=REC_ERR,-
                                     1587  5927                         P1=#T_REC_CONT
```

```
          00001D47'8F    DD  1587                          PUSHL    #T_REC_CONT
          00001EB4'EF    DF  158D                          PUSHAL   REC_ERR
          000018C8'EF    DF  1593                          PUSHAL   T_WR_BLK_ERR
                   00    DD  1599                          PUSHL    #0
                   02    DD  159B                          PUSHL    #$ER
                             159D            ;;; TEST 10, SUBTEST 10, ERROR 2
      000100D0 9F  05    FB  159D                          CALLS    #$$M, @#DS$ERRHARD
                             15A4   5928             $DS_BNCOMPLETE  40$
                03 50    E9  15A4                          BLBC     R0, 40$
                             15A7   5929             $DS_ABORT      TEST
                   50    D4  15A7                          CLRL     R0                     ; SEND A WARNING TO THE SUPERVISOR
                         04  15A9                          RET                             ; TERMINATE TEST
                             15AA   5930
                             15AA   5931 40$:        $DS_ABORT
      00010020 9F  6C    FA  15AA                          CALLG    (AP), @#DS$ABORT
                             15B1   5932                                                   ; check continue
            5C     01    D0  15B1   5933 70$:        MOVL     #1,R12                       ; load number of bytes to check
                             15B4   5934             $DS_BREAK
      00010058 9F  6E    FA  15B4                          CALLG    (SP), @#DS$BREAK
          0000213B'EF    16  15BB   5935             JSB      CHECK_PACKET                 ; go check continue
                             15C1   5936             $DS_BNCOMPLETE  90$                   ; br = no error
                24 50    E9  15C1                          BLBC     R0, 90$
                             15C4   5937             $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=DATA_ERR_CONT
          00001E4D'EF    DF  15C4                          PUSHAL   DATA_ERR_CONT
          000018C8'EF    DF  15CA                          PUSHAL   T_WR_BLK_ERR
                   00    DD  15D0                          PUSHL    #0
                   03    DD  15D2                          PUSHL    #$ER
                             15D4            ;;; TEST 10, SUBTEST 10, ERROR 3
      000100D0 9F  04    FB  15D4                          CALLS    #$$M, @#DS$ERRHARD
                             15DB   5938             $DS_BNCOMPLETE  80$
                03 50    E9  15DB                          BLBC     R0, 80$
                             15DE   5939             $DS_ABORT      TEST
                   50    D4  15DE                          CLRL     R0                     ; SEND A WARNING TO THE SUPERVISOR
                         04  15E0                          RET                             ; TERMINATE TEST
                             15E1   5940 80$:        $DS_ABORT
      00010020 9F  6C    FA  15E1                          CALLG    (AP), @#DS$ABORT
                             15E8   5941                                                   ; send data packet
      54   FFFB6C20 8F   D0  15E8   5942 90$:        MOVL     #DELAY_TIME1,R4              ; load send packet delay time
                55 93    DE  15EF   5943             MOVAL    @(R3)+,R5                    ; point to data packet
      57   00000084 8F   D0  15F2   5944             MOVL     #132,R7                      ; load transfer byte count
          0000203D'EF    16  15F9   5945             JSB      SEND_TU58                    ; send data packet
   24 00000000'EF  00    E1  15FF   5946             BBC      #V_CLOCK_FLAG,B_FLAGS,110$ ; br = no timeout error
                             1607   5947             $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=NO_READY
          00001BEB'EF    DF  1607                          PUSHAL   NO_READY
          000018C8'EF    DF  160D                          PUSHAL   T_WR_BLK_ERR
                   00    DD  1613                          PUSHL    #0
                   04    DD  1615                          PUSHL    #$ER
                             1617            ;;; TEST 10, SUBTEST 10, ERROR 4
      000100D0 9F  04    FB  1617                          CALLS    #$$M, @#DS$ERRHARD
                             161E   5948             $DS_BNCOMPLETE  100$
                03 50    E9  161E                          BLBC     R0, 100$
                             1621   5949             $DS_ABORT      TEST
                   50    D4  1621                          CLRL     R0                     ; SEND A WARNING TO THE SUPERVISOR
                         04  1623                          RET                             ; TERMINATE TEST
                             1624   5950 100$:       $DS_ABORT
          00010020 9F    6C    FA  1624                        CALLG   (AP), @#DS$ABORT
                                   162B  5951
              00000000'FF    94  162B  5952 110$:    CLRB    @A_BUFFER_BEGIN           ; clear buffer used to store continue
    FF21 00000003'EF   01    03    9D  1631  5953             ACBB    #3,#1,B_COUNT,35$         ; br = not finished
         10 00000001'EF   01    E1  163B  5954 112$:    BBC     #V_REPEAT_SEND,B_FLAGS2,115$ ; br =  not repeating
              00000003'EF    94  1643  5955             CLRB    B_COUNT                   ; initialize counter to repeat
          00000001'EF   02    8A  1649  5956             BICB2   #M_REPEAT_SEND,B_FLAGS2 ; init repeat flag to stop repeating
                      FF09    31  1650  5957             BRW     35$                       ; finish sending data
                                   1653  5958                                               ; read end packet
          54    FFE17B80 8F    D0  1653  5959 115$:    MOVL    #DELAY_TIMES,R4           ; load receive packet delay time
          55    00000000'FF    9E  165A  5960             MOVAB   @A_BUFFER_BEGIN,R5        ; load buffer address to store data
             57    55    0E    C1  1661  5961             ADDL3   #14,R5,R7                 ; load last expected address of data
              00002083'EF    16  1665  5962             JSB     RECEIVE_TU58              ; go get data from TU58
                     5A    93    DE  166B  5963             MOVAL   @(R3)+,R10                ; load address of expected end packet
          5B    00000000'FF    9E  166E  5964             MOVAB   @A_BUFFER_BEGIN,R11       ; load address of actual end packet
                     57    55    D1  1675  5965             CMPL    R5,R7                     ; check for correct number of bytes
                           31    13  1678  5966             BEQL    150$                      ; br = correct number of bytes
          00000000'EF    08    88  167A  5967             BISB2   #M_END_FLAG,B_FLAGS       ; set end error flag
                                   1681  5968 117$:    $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=REC_ERR,-
                                   1681  5969                                      P1=#T_REC_END
              00001D2E'8F    DD  1681                        PUSHL   #T_REC_END
              00001EB4'EF    DF  1687                        PUSHAL  REC_ERR
              000018C8'EF    DF  168D                        PUSHAL  T_WR_BLK_ERR
                           00    DD  1693                        PUSHL   #0
                           05    DD  1695                        PUSHL   #$ER
                                   1697        ;;; TEST 10, SUBTEST 10, ERROR 5
          000100D0 9F    05    FB  1697                        CALLS   #$$M, @#DS$ERRHARD
                                   169E  5970             $DS_BNCOMPLETE  120$
                     03 50    E9  169E                        BLBC    R0, 120$
                                   16A1  5971             $DS_ABORT        TEST
                           50    D4  16A1                        CLRL    R0                        ; SEND A WARNING TO THE SUPERVISOR
                                 04  16A3                        RET                               ; TERMINATE TEST
                                   16A4  5972
                                   16A4  5973 120$:    $DS_ABORT
          00010020 9F    6C    FA  16A4                        CALLG   (AP), @#DS$ABORT
                                   16AB  5974                                               ; check end packet
                     5C    0E    D0  16AB  5975 150$:    MOVL    #14,R12                   ; load number of bytes to check
                                   16AE  5976             $DS_BREAK
          00010058 9F    6E    FA  16AE                        CALLG   (SP), @#DS$BREAK
              0000213B'EF    16  16B5  5977             JSB     CHECK_PACKET              ; go check end packet
                                   16BB  5978             $DS_BNCOMPLETE  160$              ; br = no error
                     24 50    E9  16BB                        BLBC    R0, 160$
                                   16BE  5979             $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=DATA_ERR_END
              00001E9D'EF    DF  16BE                        PUSHAL  DATA_ERR_END
              000018C8'EF    DF  16C4                        PUSHAL  T_WR_BLK_ERR
                           00    DD  16CA                        PUSHL   #0
                           06    DD  16CC                        PUSHL   #$ER
                                   16CE        ;;; TEST 10, SUBTEST 10, ERROR 6
          000100D0 9F    04    FB  16CE                        CALLS   #$$M, @#DS$ERRHARD
                                   16D5  5980             $DS_BNCOMPLETE  155$
                     03 50    E9  16D5                        BLBC    R0, 155$
                                   16D8  5981             $DS_ABORT        TEST
                           50    D4  16D8                        CLRL    R0                        ; SEND A WARNING TO THE SUPERVISOR
                                 04  16DA                        RET                               ; TERMINATE TEST
                                   16DB  5982 155$:    $DS_ABORT
          00010020 9F    6C    FA  16DB                         CALLG   (AP), @#DS$ABORT
                                   16E2  5983                                                  ; clear buffers
               00000000'EF    94   16E2  5984 160$:     CLRB    B_FLAGS                        ; initialize flags
          58   00000000'FF    9E   16E8  5985           MOVAB   @A_BUFFER_BEGIN,R8             ; load buffer starting address
          59   00000600 8F    D0   16EF  5986           MOVL    #BUFFER_SIZE,R9                ; do all pages
               00002134'EF    16   16F6  5987           JSB     CLEAR_BUFFER                   ; go clear buffers
                                   16FC  5988                                                  ; send read command
          54   FFFB6C20 8F    D0   16FC  5989           MOVL    #DELAY_TIME1,R4                ; load send packet delay time
                    55   93   DE   1703  5990           MOVAL   @(R3)+,R5                      ; point to read command packet
                    57   0E   D0   1706  5991           MOVL    #14,R7                         ; load transfer byte count
               0000203D'EF    16   1709  5992           JSB     SEND_TU58                      ; send command packet
       24 00000000'EF    00   E1   170F  5993           BBC     #V_CLOCK_FLAG,B_FLAGS,190$ ; br = no timeout error
                                   1717  5994           $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=NO_READY
               00001BEB'EF    DF   1717                         PUSHAL  NO_READY
               000018EB'EF    DF   171D                         PUSHAL  T_RD_BLK_ERR
                         00   DD   1723                         PUSHL   #0
                         07   DD   1725                         PUSHL   #$ER
                                   1727            ;;; TEST 10, SUBTEST 10, ERROR 7
          000100D0 9F    04   FB   1727                         CALLS   #$$M, @#DS$ERRHARD
                                   172E  5995           $DS_BNCOMPLETE  180$
                      03 50   E9   172E                         BLBC    R0, 180$
                                   1731  5996           $DS_ABORT       TEST
                         50   D4   1731                         CLRL    R0                     ; SEND A WARNING TO THE SUPERVISOR
                         04   1733                              RET                            ; TERMINATE TEST
                                   1734  5997 180$:     $DS_ABORT
          00010020 9F    6C   FA   1734                         CALLG   (AP), @#DS$ABORT
                                   173B  5998                                                  ; read data and end packets
          54   FFF3CB00 8F    D0   173B  5999 190$:     MOVL    #DELAY_TIME3,R4                ; load receive packet delay time
          55   00000000'FF    9E   1742  6000           MOVAB   @A_BUFFER_BEGIN,R5             ; load buffer address to store data
     57   55   00000210 8F    C1   1749  6001           ADDL3   #528,R5,R7                     ; load last expected address of data
       07 00000001'EF    02   E1   1751  6002           BBC     #V_REPEAT_CHECK,B_FLAGS2,195$ ; br = not repeating
          57   00000210 8F    C0   1759  6003           ADDL2   #528,R7                        ; must double buffer if repeating
                    57   0E   C0   1760  6004 195$:     ADDL2   #14,R7                         ; make last address include end packet
               00002083'EF    16   1763  6005           JSB     RECEIVE_TU58                   ; go get data from TU58
          5B   00000000'FF    9E   1769  6006           MOVAB   @A_BUFFER_BEGIN,R11            ; load address of actual data
                    5A   93   DE   1770  6007           MOVAL   @(R3)+,R10                     ; load address of expected data
                    57   55   D1   1773  6008           CMPL    R5,R7                          ; check for correct number of bytes
                         31   13   1776  6009           BEQL    230$                           ; br = correct number of bytes
          00000000'EF    04   88   1778  6010           BISB2   #M_DATA_FLAG,B_FLAGS           ; set data error flag
                                   177F  6011 197$:     $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=REC_ERR,-
                                   177F  6012                            P1=#T_REC_DATA
               00001D68'8F    DD   177F                         PUSHL   #T_REC_DATA
               00001EB4'EF    DF   1785                         PUSHAL  REC_ERR
               000018EB'EF    DF   178B                         PUSHAL  T_RD_BLK_ERR
                         00   DD   1791                         PUSHL   #0
                         08   DD   1793                         PUSHL   #$ER
                                   1795            ;;; TEST 10, SUBTEST 10, ERROR 8
          000100D0 9F    05   FB   1795                         CALLS   #$$M, @#DS$ERRHARD
                                   179C  6013           $DS_BNCOMPLETE  200$
                      03 50   E9   179C                         BLBC    R0, 200$
                                   179F  6014           $DS_ABORT       TEST
                         50   D4   179F                         CLRL    R0                     ; SEND A WARNING TO THE SUPERVISOR
                              04   17A1                         RET                            ; TERMINATE TEST
                                   17A2  6015
                                   17A2  6016 200$:     $DS_ABORT
```

```
         00010020 9F    6C   FA  17A2                           CALLG   (AP), @#DS$ABORT
                                 17A9  6017                                                     ; check data packets
              00000006'EF    D4  17A9  6018 230$:    CLRL       L_COUNT                         ; clear counter to check 4 packets
         5C   00000084 8F    D0  17AF  6019          MOVL       #132,R12                        ; load number of bytes to check
                                 17B6  6020 240$:    $DS_BREAK
         00010058 9F    6E   FA  17B6                           CALLG   (SP), @#DS$BREAK
              0000213B'EF    16  17BD  6021          JSB        CHECK_PACKET                    ; go check data packet
                                 17C3  6022          $DS_BNCOMPLETE  260$                       ; br = no error
                    24 50    E9  17C3                           BLBC    R0, 260$
                                 17C6  6023          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=DATA_ERR_DATA
              00001E75'EF    DF  17C6                           PUSHAL  DATA_ERR_DATA
              000018EB'EF    DF  17CC                           PUSHAL  T_RD_BLK_ERR
                       00    DD  17D2                           PUSHL   #0
                       09    DD  17D4                           PUSHL   #$ER
                                 17D6              ;;; TEST 10, SUBTEST 10, ERROR 9
         000100D0 9F    04   FB  17D6                           CALLS   #$$M, @#DS$ERRHARD
                                 17DD  6024          $DS_BNCOMPLETE  250$
                    03 50    E9  17DD                           BLBC    R0, 250$
                                 17E0  6025          $DS_ABORT          TEST
                       50    D4  17E0                           CLRL    R0                      ; SEND A WARNING TO THE SUPERVISOR
                       04    04  17E2                           RET                             ; TERMINATE TEST
                                 17E3  6026 250$:    $DS_ABORT
         00010020 9F    6C   FA  17E3                           CALLG   (AP), @#DS$ABORT
         5B   00000084 8F    C0  17EA  6027 260$:    ADDL2      #132,R11                        ; adjust pointer to next packet
                    5A    93 DE  17F1  6028          MOVAL      @(R3)+,R10                      ; load address of expected data
      BA 00000006'EF    03   F3  17F4  6029          AOBLEQ     #3,L_COUNT,240$                 ; br = not finished checking data
      0F 00000001'EF    02   E1  17FC  6030          BBC        #V_REPEAT_CHECK,B_FLAGS2,265$ ; br = not repeating
              00000006'EF    D4  1804  6031          CLRL       L_COUNT                         ; initialize check counter
              00000001'EF    94  180A  6032          CLRB       B_FLAGS2                        ; clear repeat flag to stop repeating
                     FFA3    31  1810  6033          BRW        240$                            ; check rest of data packets
                                 1813  6034                                                     ; check end packet
               5C     0E     D0  1813  6035 265$:    MOVL       #14,R12                         ; load number of bytes to check
                                 1816  6036          $DS_BREAK
         00010058 9F    6E   FA  1816                           CALLG   (SP), @#DS$BREAK
              0000213B'EF    16  181D  6037          JSB        CHECK_PACKET                    ; go check end packet
                                 1823  6038          $DS_BNCOMPLETE  275$                       ; br = no error
                    24 50    E9  1823                           BLBC    R0, 275$
                                 1826  6039          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=DATA_ERR_END
              00001E9D'EF    DF  1826                           PUSHAL  DATA_ERR_END
              000018EB'EF    DF  182C                           PUSHAL  T_RD_BLK_ERR
                       00    DD  1832                           PUSHL   #0
                       0A    DD  1834                           PUSHL   #$ER
                                 1836              ;;; TEST 10, SUBTEST 10, ERROR 10
         000100D0 9F    04   FB  1836                           CALLS   #$$M, @#DS$ERRHARD
                                 183D  6040          $DS_BNCOMPLETE  270$
                    03 50    E9  183D                           BLBC    R0, 270$
                                 1840  6041          $DS_ABORT          TEST
                       50    D4  1840                           CLRL    R0                      ; SEND A WARNING TO THE SUPERVISOR
                       04    04  1842                           RET                             ; TERMINATE TEST
                                 1843  6042 270$:    $DS_ABORT
         00010020 9F    6C   FA  1843                           CALLG   (AP), @#DS$ABORT
  FCA9 00000004'EF    01    04   9D  184A  6043 275$:    ACBB       #4,#1,B_COUNT1,10$              ; br = not finished executing test
                                 1854  6044
                                 1854  6045
                                 1854  6046          $DS_ENDSUB
```

```
                                   1854        T10_S10_X:
     00010038 9F    00000090'EF    FA  1854                      CALLG    $S$, @#DS$ENDSUB
                                   185F   6047
                                   185F   6048         $DS_PAGE
```

```
185F                 .SBTTL  SUBTEST 10.11:   RECORD WRITE - READ
185F       ;+
185F       ; SUBTEST DESCRIPTION:
185F       ;
185F       ;        This subtest sends the TU58 controller a series of command packets
185F       ;        that will cause head switches, forward seeks, reverse seeks and
185F       ;        seeks with head switches durring read and write operations on
185F       ;        records. All data written is read and checked. All end packets
185F       ;        are checked as the subtest progresses. If any errors occur they are
185F       ;        reported.
185F       ;
185F       ;
185F       ; SUBTEST STEPS:
185F       ;
185F       ;        SUBTEST RECORD WRITE - READ
185F       ;        :  DO for count = 1 to 5
185F       ;        :  :  Point to write command packet
185F       ;        :  :  Clear buffer
185F       ;        :  :  REPEAT
185F       ;        :  :  :  Send command packet byte at a time
185F       ;        :  :  :  Set timer
185F       ;        :  :  :  IF timer expired AND no response
185F       ;        :  :  :  :  THEN
185F       ;        :  :  :  :  Report no response error
185F       ;        :  :  :  :  Exit test
185F       ;        :  :  :  ENDIF
185F       ;        :  :  : UNTIL all bytes sent
185F       ;        :  :  ENDREPEAT
185F       ;        :  :  DO for count = 1 to 2
185F       ;        :  :  :  Wait for continue
185F       ;        :  :  :  Set timer
185F       ;        :  :  :  IF timer expired AND no continue
185F       ;        :  :  :  :  THEN
185F       ;        :  :  :  :  Report no continue error
185F       ;        :  :  :  :  Exit test
185F       ;        :  :  :  ENDIF
185F       ;        :  :  :  Point to data packet
185F       ;        :  :  :  REPEAT
185F       ;        :  :  :  :  Send data packet byte at a time
185F       ;        :  :  :  :  Set timer
185F       ;        :  :  :  :  IF timer expired AND no response
185F       ;        :  :  :  :  :  THEN
185F       ;        :  :  :  :  :  Report no response error
185F       ;        :  :  :  :  :  Exit test
185F       ;        :  :  :  :  ENDIF
185F       ;        :  :  :  : UNTIL all bytes sent
185F       ;        :  :  :  ENDREPEAT
185F       ;        :  :  ENDDO
185F       ;        :  :  REPEAT
185F       ;        :  :  :  Set timer
185F       ;        :  :  :  IF timer expired AND done set
185F       ;        :  :  :  :  THEN
185F       ;        :  :  :  :  :  Read byte
185F       ;        :  :  :  ENDIF
185F       ;        :  :  : UNTIL timer expired AND done clear
```

```
185F  6105 ;        :  :  ENDREPEAT
185F  6106 ;        :  :  IF expected byte count NEQ actual byte count
185F  6107 ;        :  :  :  THEN
185F  6108 ;        :  :  :  :  Report error
185F  6109 ;        :  :  :  :  Exit test
185F  6110 ;        :  :  ENDIF
185F  6111 ;        :  :  REPEAT
185F  6112 ;        :  :  :  Point to expected end packet
185F  6113 ;        :  :  :  IF expected end data NEQ received end data
185F  6114 ;        :  :  :  :  THEN
185F  6115 ;        :  :  :  :  Report end packet data error
185F  6116 ;        :  :  :  ENDIF
185F  6117 ;        :  :  : UNTIL all data checked
185F  6118 ;        :  :  ENDREPEAT
185F  6119 ;        :  :  Point to read command packet
185F  6120 ;        :  :  REPEAT
185F  6121 ;        :  :  :  Send command packet byte at a time
185F  6122 ;        :  :  :  Set timer
185F  6123 ;        :  :  :  IF timer expired and no response
185F  6124 ;        :  :  :  :  THEN
185F  6125 ;        :  :  :  :  Report no response error
185F  6126 ;        :  :  :  :  Exit test
185F  6127 ;        :  :  :  ENDIF
185F  6128 ;        :  :  : UNTIL all bytes sent
185F  6129 ;        :  :  ENDREPEAT
185F  6130 ;        :  :  REPEAT
185F  6131 ;        :  :  :  Set timer
185F  6132 ;        :  :  :  IF timer expired AND done set
185F  6133 ;        :  :  :  :  THEN
185F  6134 ;        :  :  :  :  :  Read byte
185F  6135 ;        :  :  :  ENDIF
185F  6136 ;        :  :  : UNTIL timer expired AND done clear
185F  6137 ;        :  :  ENDREPEAT
185F  6138 ;        :  :  IF expected byte count NEQ actual byte count
185F  6139 ;        :  :  :  THEN
185F  6140 ;        :  :  :  :  Report error
185F  6141 ;        :  :  :  :  Exit test
185F  6142 ;        :  :  ENDIF
185F  6143 ;        :  :  REPEAT
185F  6144 ;        :  :  :  Point to expected data packet
185F  6145 ;        :  :  :  REPEAT
185F  6146 ;        :  :  :  :  IF data expected NEQ data received
185F  6147 ;        :  :  :  :  :  THEN
185F  6148 ;        :  :  :  :  :  Report data error
185F  6149 ;        :  :  :  :  :  Exit subtest
185F  6150 ;        :  :  :  :  ENDIF
185F  6151 ;        :  :  :  :  Point to next byte of data
185F  6152 ;        :  :  :  : UNTIL all data in packet checked
185F  6153 ;        :  :  :  ENDREPEAT
185F  6154 ;        :  :  : UNTIL 4 data packets checked
185F  6155 ;        :  :  ENDREPEAT
185F  6156 ;        :  :  Point to expected end packet
185F  6157 ;        :  :  REPEAT
185F  6158 ;        :  :  :  IF data expected NEQ data received
185F  6159 ;        :  :  :  :  THEN
```

```
                              185F  6160 ;          :  :  :  :  Report data error
                              185F  6161 ;          :  :  :  :  Exit subtest
                              185F  6162 ;          :  :  :  ENDIF
                              185F  6163 ;          :  :  : UNTIL all end packet bytes checked
                              185F  6164 ;          :  :  ENDREPEAT
                              185F  6165 ;          :  ENDDO
                              185F  6166 ;          ENDSUBTEST RECORD WRITE - READ
                              185F  6167 ;
                              185F  6168 ;
                              185F  6169 ;.page
                              185F  6170 ;-
                              185F  6171            $DS_BGNSUB
                              185F       T10_S11::
    00010030 9F   0000009C'EF   FA  185F                    CALLG    $S$, a#DS$BGNSUB
                  00000001'EF   94  186A  6172      CLRB    B_FLAGS2                    ; initialize repeat flag
           00000001'EF   06     88  1870  6173      BISB2   #<M_REPEAT_SEND+M_REPEAT_CHECK>,B_FLAGS2
                  00000004'EF   94  1877  6174      CLRB    B_COUNT1                    ; initialize command counter
           53     000014E3'EF   9E  187D  6175      MOVAB   COMMAND_TBL_4,R3            ; load pointer to command table
                                    1884  6176                                          ; clear buffers
                  00000000'EF   94  1884  6177 10$: CLRB    B_FLAGS                     ; initialize flags
           58     00000000'FF   9E  188A  6178      MOVAB   aA_BUFFER_BEGIN,R8          ; load buffer starting address
           59     00000600 8F   D0  1891  6179      MOVL    #BUFFER_SIZE,R9             ; do all pages
                  00002134'EF   16  1898  6180      JSB     CLEAR_BUFFER                ; go clear buffers
                                    189E  6181                                          ; send write command
           54     FFF3CB00 8F   D0  189E  6182      MOVL    #DELAY_TIME3,R4             ; load send packet delay time
                        55  93  DE  18A5  6183      MOVAL   a(R3)+,R5                   ; point to write command packet
                        57  0E  D0  18A8  6184      MOVL    #14,R7                      ; load transfer byte count
                  0000203D'EF   16  18AB  6185      JSB     SEND_TU58                   ; send command packet
        24 00000000'EF   00     E1  18B1  6186      BBC     #V_CLOCK_FLAG,B_FLAGS,30$ ; br = no timeout error
                                    18B9  6187      $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=NO_READY
                  00001BEB'EF   DF  18B9                    PUSHAL  NO_READY
                  0000190D'EF   DF  18BF                    PUSHAL  T_WR_REC_ERR
                           00   DD  18C5                    PUSHL   #0
                           01   DD  18C7                    PUSHL   #$ER
                                    18C9       ;;; TEST 10, SUBTEST 11, ERROR 1
           000100D0 9F    04    FB  18C9                    CALLS   #$$M, a#DS$ERRHARD
                                    18D0  6188      $DS_BNCOMPLETE  20$
                     03 50      E9  18D0                    BLBC    R0, 20$
                                    18D3  6189      $DS_ABORT       TEST
                           50   D4  18D3                    CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                           04   18D5                        RET                         ; TERMINATE TEST
                                    18D6  6190 20$: $DS_ABORT
           00010020 9F    6C    FA  18D6                    CALLG   (AP), a#DS$ABORT
                                    18DD  6191                                          ; read continue
           54     FF7E4CE0 8F   D0  18DD  6192 30$: MOVL    #DELAY_TIME7,R4             ; load receive packet delay time
           55     00000000'FF   9E  18E4  6193      MOVAB   aA_BUFFER_BEGIN,R5          ; load buffer address to store data
               57    55   01    C1  18EB  6194      ADDL3   #1,R5,R7                    ; load last expected address of data
                  00002083'EF   16  18EF  6195      JSB     RECEIVE_TU58                ; go get data from TU58
           5A     00000023'EF   9E  18F5  6196      MOVAB   B_CONTINUE,R10              ; load address of expected data
           5B     00000000'FF   9E  18FC  6197      MOVAB   aA_BUFFER_BEGIN,R11         ; load address of actual data
                     57   55    D1  1903  6198      CMPL    R5,R7                       ; check for correct number of bytes
                          2A    13  1906  6199      BEQL    70$                         ; br = correct number of bytes
                                    1908  6200      $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=REC_ERR,-
                                    1908  6201                      P1=#T_REC_CONT
                  00001D47'8F   DD  1908                    PUSHL   #T_REC_CONT
```

```
              00001EB4'EF   DF  190E                     PUSHAL  REC_ERR
              0000190D'EF   DF  1914                     PUSHAL  T_WR_REC_ERR
                       00   DD  191A                     PUSHL   #0
                       02   DD  191C                     PUSHL   #$ER
                                191E         ;;; TEST 10, SUBTEST 11, ERROR 2
          000100D0 9F  05   FB  191E                     CALLS   #$$M, @#DS$ERRHARD
                                1925  6202         $DS_BNCOMPLETE  40$
                    03 50   E9  1925                     BLBC    R0, 40$
                                1928  6203         $DS_ABORT       TEST
                       50   D4  1928                     CLRL    R0                 ; SEND A WARNING TO THE SUPERVISOR
                       04       192A                     RET                        ; TERMINATE TEST
                                192B  6204
                                192B  6205 40$:    $DS_ABORT
          00010020 9F  6C   FA  192B                     CALLG   (AP), @#DS$ABORT
                                1932  6206                                          ; check continue
                 5C    01   D0  1932  6207 70$:    MOVL    #1,R12                   ; load number of bytes to check
                                1935  6208         $DS_BREAK
          00010058 9F  6E   FA  1935                     CALLG   (SP), @#DS$BREAK
              0000213B'EF   16  193C  6209         JSB     CHECK_PACKET             ; go check continue
                                1942  6210         $DS_BNCOMPLETE  90$              ; br = no error
                    24 50   E9  1942                     BLBC    R0, 90$
                                1945  6211         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=DATA_ERR_CONT
              00001E4D'EF   DF  1945                     PUSHAL  DATA_ERR_CONT
              0000190D'EF   DF  194B                     PUSHAL  T_WR_REC_ERR
                       00   DD  1951                     PUSHL   #0
                       03   DD  1953                     PUSHL   #$ER
                                1955         ;;; TEST 10, SUBTEST 11, ERROR 3
          000100D0 9F  04   FB  1955                     CALLS   #$$M, @#DS$ERRHARD
                                195C  6212         $DS_BNCOMPLETE  80$
                    03 50   E9  195C                     BLBC    R0, 80$
                                195F  6213         $DS_ABORT       TEST
                       50   D4  195F                     CLRL    R0                 ; SEND A WARNING TO THE SUPERVISOR
                       04       1961                     RET                        ; TERMINATE TEST
                                1962  6214 80$:    $DS_ABORT
          00010020 9F  6C   FA  1962                     CALLG   (AP), @#DS$ABORT
                                1969  6215                                          ; send data packet
      54    FFFB6C20 8F     D0  1969  6216 90$:    MOVL    #DELAY_TIME1,R4          ; load send packet delay time
                 55    93   DE  1970  6217         MOVAL   @(R3)+,R5                ; point to data packet
      57    00000084 8F     D0  1973  6218         MOVL    #132,R7                  ; load transfer byte count
              0000203D'EF   16  197A  6219         JSB     SEND_TUS8                ; send data packet
   24 00000000'EF      00   E1  1980  6220         BBC     #V_CLOCK_FLAG,B_FLAGS,110$ ; br = no timeout error
                                1988  6221         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=NO_READY
              00001BEB'EF   DF  1988                     PUSHAL  NO_READY
              0000190D'EF   DF  198E                     PUSHAL  T_WR_REC_ERR
                       00   DD  1994                     PUSHL   #0
                       04   DD  1996                     PUSHL   #$ER
                                1998         ;;; TEST 10, SUBTEST 11, ERROR 4
          000100D0 9F  04   FB  1998                     CALLS   #$$M, @#DS$ERRHARD
                                199F  6222         $DS_BNCOMPLETE  100$
                    03 50   E9  199F                     BLBC    R0, 100$
                                19A2  6223         $DS_ABORT       TEST
                       50   D4  19A2                     CLRL    R0                 ; SEND A WARNING TO THE SUPERVISOR
                       04       19A4                     RET                        ; TERMINATE TEST
                                19A5  6224 100$:   $DS_ABORT
          00010020 9F  6C   FA  19A5                     CALLG   (AP), @#DS$ABORT
```

```
                                  19AC  6225
          00000000'FF     94      19AC  6226 110$:     CLRB     @A_BUFFER_BEGIN         ; clear buffer used to store continue
     0A 00000001'EF     01      E1    19B2  6227            BBC      #V_REPEAT_SEND,B_FLAGS2,115$ ; br =  not repeating
          00000001'EF     02      8A      19BA  6228            BICB2    #M_REPEAT_SEND,B_FLAGS2 ; clear repeat flag to stop repeating
                    FF19     31      19C1  6229            BRW      30$                     ; finish sending data
                                  19C4  6230                                              ; read end packet
     54   FF7E4CE0 8F     D0      19C4  6231 115$:     MOVL     #DELAY_TIME7,R4         ; load receive packet delay time
     55   00000000'FF     9E      19CB  6232            MOVAB    @A_BUFFER_BEGIN,R5      ; load buffer address to store data
        57    55    0E     C1      19D2  6233            ADDL3    #14,R5,R7               ; load last expected address of data
          00002083'EF     16      19D6  6234            JSB      RECEIVE_TU58            ; go get data from TU58
              5A    93     DE      19DC  6235            MOVAL    @(R3)+,R10              ; load address of expected end packet
     5B   00000000'FF     9E      19DF  6236            MOVAB    @A_BUFFER_BEGIN,R11     ; load address of actual end packet
              57    55     D1      19E6  6237            CMPL     R5,R7                   ; check for correct number of bytes
                    31     13      19E9  6238            BEQL     150$                    ; br = correct number of bytes
          00000000'EF     08      88      19EB  6239            BISB2    #M_END_FLAG,B_FLAGS     ; set end error flag
                                  19F2  6240 117$:     $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=REC_ERR,-
                                  19F2  6241                             P1=#T_REC_END
          00001D2E'8F     DD      19F2                       PUSHL    #T_REC_END
          00001EB4'EF     DF      19F8                       PUSHAL   REC_ERR
          0000190D'EF     DF      19FE                       PUSHAL   T_WR_REC_ERR
                    00     DD      1A04                       PUSHL    #0
                    05     DD      1A06                       PUSHL    #$ER
                                  1A08       ;;; TEST 10, SUBTEST 11, ERROR 5
          000100D0 9F     05     FB      1A08                       CALLS    #$$M, @#DS$ERRHARD
                                  1A0F  6242            $DS_BNCOMPLETE  120$
                 03 50     E9      1A0F                       BLBC     R0, 120$
                                  1A12  6243            $DS_ABORT       TEST
                    50     D4      1A12                       CLRL     R0                ; SEND A WARNING TO THE SUPERVISOR
                           04      1A14                       RET                        ; TERMINATE TEST
                                  1A15  6244
                                  1A15  6245 120$:     $DS_ABORT
          00010020 9F     6C     FA      1A15                       CALLG    (AP), @#DS$ABORT
                                  1A1C  6246                                              ; check end packet
              5C    0E     D0      1A1C  6247 150$:     MOVL     #14,R12                 ; load number of bytes to check
                                  1A1F  6248            $DS_BREAK
          00010058 9F     6E     FA      1A1F                       CALLG    (SP), @#DS$BREAK
          0000213B'EF     16      1A26  6249            JSB      CHECK_PACKET            ; go check end packet
                                  1A2C  6250            $DS_BNCOMPLETE  160$             ; br = no error
                 21 50     E9      1A2C                       BLBC     R0, 160$
                                  1A2F  6251            $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=DATA_ERR_END
          00001E9D'EF     DF      1A2F                       PUSHAL   DATA_ERR_END
          0000190D'EF     DF      1A35                       PUSHAL   T_WR_REC_ERR
                    00     DD      1A3B                       PUSHL    #0
                    06     DD      1A3D                       PUSHL    #$ER
                                  1A3F       ;;; TEST 10, SUBTEST 11, ERROR 6
          000100D0 9F     04     FB      1A3F                       CALLS    #$$M, @#DS$ERRHARD
                                  1A46  6252            $DS_BCOMPLETE   160$             ; br = error, but continue
                 07 50     E8      1A46                       BLBS     R0, 160$
                                  1A49  6253            $DS_ABORT
          00010020 9F     6C     FA      1A49                       CALLG    (AP), @#DS$ABORT
                                  1A50  6254                                              ; clear buffers
          00000000'EF     94      1A50  6255 160$:     CLRB     B_FLAGS                 ; initialize flags
     58   00000000'FF     9E      1A56  6256            MOVAB    @A_BUFFER_BEGIN,R8      ; load buffer starting address
     59   00000600 8F     D0      1A5D  6257            MOVL     #BUFFER_SIZE,R9         ; do all pages
          00002134'EF     16      1A64  6258            JSB      CLEAR_BUFFER            ; go clear buffers
```

```
                                     1A6A  6259                                             ; send read command
              54    FFFB6C20 8F     D0  1A6A  6260        MOVL    #DELAY_TIME1,R4         ; load send packet delay time
                       55   93     DE  1A71  6261        MOVAL   @(R3)+,R5               ; point to read command packet
                       57   0E     D0  1A74  6262        MOVL    #14,R7                  ; load transfer byte count
                   0000203D'EF     16  1A77  6263        JSB     SEND_TU58               ; send command packet
           24 00000000'EF   00     E1  1A7D  6264        BBC     #V_CLOCK_FLAG,B_FLAGS,190$ ; br = no timeout error
                                     1A85  6265        $DS_ERRHARD_S  ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=NO_READY
                   00001BEB'EF     DF  1A85                        PUSHAL  NO_READY
                   00001931'EF     DF  1A8B                        PUSHAL  T_RD_REC_ERR
                            00     DD  1A91                        PUSHL   #0
                            07     DD  1A93                        PUSHL   #$ER
                                     1A95       ;;; TEST 10, SUBTEST 11, ERROR 7
              000100D0 9F   04     FB  1A95                        CALLS   #$$M, @#DS$ERRHARD
                                     1A9C  6266        $DS_BNCOMPLETE  180$
                         03 50     E9  1A9C                        BLBC    R0, 180$
                                     1A9F  6267        $DS_ABORT       TEST
                            50     D4  1A9F                        CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                   04  1AA1                        RET                     ; TERMINATE TEST
                                     1AA2  6268 180$:  $DS_ABORT
              00010020 9F   6C     FA  1AA2                        CALLG   (AP), @#DS$ABORT
                                     1AA9  6269                                             ; read data and end packets
              54    FFF3CB00 8F     D0  1AA9  6270 190$:  MOVL    #DELAY_TIME3,R4         ; load receive packet delay time
              55    00000000'FF     9E  1AB0  6271        MOVAB   @A_BUFFER_BEGIN,R5      ; load buffer address to store data
         57   55    00000084 8F     C1  1AB7  6272        ADDL3   #132,R5,R7              ; load last expected address of data
           07 00000001'EF   02     E1  1ABF  6273        BBC     #V_REPEAT_CHECK,B_FLAGS2,195$ ; br = not repeating
              57    00000084 8F     C0  1AC7  6274        ADDL2   #132,R7                 ; must double buffer if repeating
                       57   0E     C0  1ACE  6275 195$:  ADDL2   #14,R7                  ; add end packet to address
                   00002083'EF     16  1AD1  6276        JSB     RECEIVE_TU58            ; go get data from TU58
              5B    00000000'FF     9E  1AD7  6277        MOVAB   @A_BUFFER_BEGIN,R11     ; load address of buffer
                       5A   93     DE  1ADE  6278        MOVAL   @(R3)+,R10              ; load address of expected data
                       57   55     D1  1AE1  6279        CMPL    R5,R7                   ; check for correct number of bytes
                            31     13  1AE4  6280        BEQL    230$                    ; br = correct number of bytes
              00000000'EF   04     88  1AE6  6281        BISB2   #M_DATA_FLAG,B_FLAGS    ; set data error flag
                                     1AED  6282 197$:  $DS_ERRHARD_S  ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=REC_ERR,-
                                     1AED  6283                        P1=#T_REC_DATA
                   00001D68'8F     DD  1AED                        PUSHL   #T_REC_DATA
                   00001EB4'EF     DF  1AF3                        PUSHAL  REC_ERR
                   00001931'EF     DF  1AF9                        PUSHAL  T_RD_REC_ERR
                            00     DD  1AFF                        PUSHL   #0
                            08     DD  1B01                        PUSHL   #$ER
                                     1B03       ;;; TEST 10, SUBTEST 11, ERROR 8
              000100D0 9F   05     FB  1B03                        CALLS   #$$M, @#DS$ERRHARD
                                     1B0A  6284        $DS_BNCOMPLETE  200$
                         03 50     E9  1B0A                        BLBC    R0, 200$
                                     1B0D  6285        $DS_ABORT       TEST
                            50     D4  1B0D                        CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                   04  1B0F                        RET                     ; TERMINATE TEST
                                     1B10  6286
                                     1B10  6287 200$:  $DS_ABORT
              00010020 9F   6C     FA  1B10                        CALLG   (AP), @#DS$ABORT
                                     1B17  6288                                             ; check data packets
              5C    00000084 8F     D0  1B17  6289 230$:  MOVL    #132,R12                ; load number of bytes to check
                                     1B1E  6290 240$:  $DS_BREAK
              00010058 9F   6E     FA  1B1E                        CALLG   (SP), @#DS$BREAK
                   0000213B'EF     16  1B25  6291        JSB     CHECK_PACKET            ; go check data packet
```

```
                              1B2B  6292          $DS_BNCOMPLETE  260$              ; br = no error
                  24 50   E9  1B2B                     BLBC    R0, 260$
                              1B2E  6293          $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=DATA_ERR_DATA
            00001E75'EF   DF  1B2E                     PUSHAL  DATA_ERR_DATA
            00001931'EF   DF  1B34                     PUSHAL  T_RD_REC_ERR
                     00   DD  1B3A                     PUSHL   #0
                     09   DD  1B3C                     PUSHL   #$ER
                              1B3E         ;;; TEST 10, SUBTEST 11, ERROR 9
       000100D0 9F   04   FB  1B3E                     CALLS   #$$M, @#DS$ERRHARD
                              1B45  6294          $DS_BNCOMPLETE  250$
                  03 50   E9  1B45                     BLBC    R0, 250$
                              1B48  6295          $DS_ABORT       TEST
                     50   D4  1B48                     CLRL    R0                ; SEND A WARNING TO THE SUPERVISOR
                          04  1B4A                     RET                       ; TERMINATE TEST
                              1B4B  6296 250$:    $DS_ABORT
       00010020 9F   6C   FA  1B4B                     CALLG   (AP), @#DS$ABORT
       5B   00000084 8F   C0  1B52  6297 260$:    ADDL2   #132,R11                 ; adjust pointer to next packet
    0C 00000001'EF   02   E1  1B59  6298          BBC     #V_REPEAT_CHECK,B_FLAGS2,265$ ; br = not repeating
            00000001'EF   94  1B61  6299          CLRB    B_FLAGS2                 ; clear repeat flag to stop repeating
                  5A 93   DE  1B67  6300          MOVAL   @(R3)+,R10               ; load address of expected data
                   FFB1   31  1B6A  6301          BRW     240$                     ; check rest of data packets
                              1B6D  6302                                           ; check end packet
                  5A 93   DE  1B6D  6303 265$:    MOVAL   @(R3)+,R10               ; load address of expected end packet
                  5C 0E   D0  1B70  6304          MOVL    #14,R12                  ; load number of bytes to check
                              1B73  6305          $DS_BREAK
       00010058 9F   6E   FA  1B73                     CALLG   (SP), @#DS$BREAK
            0000213B'EF   16  1B7A  6306          JSB     CHECK_PACKET             ; go check end packet
                              1B80  6307          $DS_BNCOMPLETE  275$              ; br = no error
                  24 50   E9  1B80                     BLBC    R0, 275$
                              1B83  6308          $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=DATA_ERR_END
            00001E9D'EF   DF  1B83                     PUSHAL  DATA_ERR_END
            00001931'EF   DF  1B89                     PUSHAL  T_RD_REC_ERR
                     00   DD  1B8F                     PUSHL   #0
                     0A   DD  1B91                     PUSHL   #$ER
                              1B93         ;;; TEST 10, SUBTEST 11, ERROR 10
       000100D0 9F   04   FB  1B93                     CALLS   #$$M, @#DS$ERRHARD
                              1B9A  6309          $DS_BNCOMPLETE  270$
                  03 50   E9  1B9A                     BLBC    R0, 270$
                              1B9D  6310          $DS_ABORT       TEST
                     50   D4  1B9D                     CLRL    R0                ; SEND A WARNING TO THE SUPERVISOR
                          04  1B9F                     RET                       ; TERMINATE TEST
                              1BA0  6311 270$:    $DS_ABORT
       00010020 9F   6C   FA  1BA0                     CALLG   (AP), @#DS$ABORT
  FCD3 00000004'EF   01   04   9D  1BA7  6312 275$:    ACBB    #4,#1,B_COUNT1,10$       ; br = not finished executing test
                              1BB1  6313
                              1BB1  6314
                              1BB1  6315          $DS_ENDSUB
                              1BB1         T10_S11_X:
  00010038 9F   0000009C'EF   FA  1BB1                     CALLG   $S$, @#DS$ENDSUB
                              1BBC  6316          $DS_PAGE
```

```
1BBC              .SBTTL  SUBTEST 10.12:   WRITE RECORD INTERRUPT
1BBC  ;+
1BBC  ; SUBTEST DESCRIPTION:
1BBC  ;
1BBC  ;       This subtest sends the TU58 a command packet to write one record
1BBC  ;       of 128 bytes in interrupt mode. When an interrupt occurs the write service-
1BBC  ;       ice routine sends the next byte. A timeout occurs after sufficient time
1BBC  ;       has passed to send a full packet, at which time an end packet is
1BBC  ;       read and checked. An error is reported if an interrupt fails to occur
1BBC  ;       before the timer expires and the full packet has not been sent. The data
1BBC  ;       packet sent contains alternating bytes of all one's and all zero's.
1BBC  ;       The data will be read and checked for errors in the next subtest.
1BBC  ;
1BBC  ;
1BBC  ; SUBTEST STEPS:
1BBC  ;
1BBC  ;       SUBTEST WRITE RECORD INTERRUPT
1BBC  ;       :  Clear buffer
1BBC  ;       :  Point to write command packet
1BBC  ;       :  REPEAT
1BBC  ;       :  :  Send command packet byte at a time
1BBC  ;       :  :  Set timer
1BBC  ;       :  :  IF timer expired AND no response
1BBC  ;       :  :  :  THEN
1BBC  ;       :  :  :  Report no response error
1BBC  ;       :  :  :  Exit test
1BBC  ;       :  :  ENDIF
1BBC  ;       :  : UNTIL all bytes sent
1BBC  ;       :  ENDREPEAT
1BBC  ;       :  Wait for continue
1BBC  ;       :  Set timer
1BBC  ;       :  IF timer expired AND no continue
1BBC  ;       :  :  THEN
1BBC  ;       :  :  Report no continue error
1BBC  ;       :  :  Exit test
1BBC  ;       :  ENDIF
1BBC  ;       :  Point to data packet
1BBC  ;       :  Enable send interrupt
1BBC  ;       :  REPEAT
1BBC  ;       :  :  Send data packet byte at a time
1BBC  ;       :  :  Set timer
1BBC  ;       :  :  IF timer expired AND no response
1BBC  ;       :  :  :  THEN
1BBC  ;       :  :  :  :  DISABLE send interrupt
1BBC  ;       :  :  :  :  Report no response error
1BBC  ;       :  :  :  :  Exit test
1BBC  ;       :  :  ENDIF
1BBC  ;       :  : UNTIL all bytes sent
1BBC  ;       :  ENDREPEAT
1BBC  ;       :  DISABLE send interrupt
1BBC  ;       :  REPEAT
1BBC  ;       :  :  Set timer
1BBC  ;       :  :  IF timer expired AND done set
1BBC  ;       :  :  :  THEN
1BBC  ;       :  :  :  :  Read byte
```

```
                                      1BBC  6373 ;       :  :  ENDIF
                                      1BBC  6374 ;       :  : UNTIL timer expired AND done clear
                                      1BBC  6375 ;       :  ENDREPEAT
                                      1BBC  6376 ;       :  IF expected byte count NEQ actual byte count
                                      1BBC  6377 ;       :  :  THEN
                                      1BBC  6378 ;       :  :  :  Report error
                                      1BBC  6379 ;       :  :  :  Exit test
                                      1BBC  6380 ;       :  ENDIF
                                      1BBC  6381 ;       :  REPEAT
                                      1BBC  6382 ;       :  :  Point to expected end packet
                                      1BBC  6383 ;       :  :  IF expected end packet data NEQ received end packet data
                                      1BBC  6384 ;       :  :  :  THEN
                                      1BBC  6385 ;       :  :  :  Report end packet data error
                                      1BBC  6386 ;       :  :  ENDIF
                                      1BBC  6387 ;       :  : UNTIL all data checked
                                      1BBC  6388 ;       :  ENDREPEAT
                                      1BBC  6389 ;       ENDSUBTEST WRITE RECORD INTERRUPT
                                      1BBC  6390 ;
                                      1BBC  6391 ;
                                      1BBC  6392 ;
                                      1BBC  6393 ;.page
                                      1BBC  6394 ;-
                                      1BBC  6395
                                      1BBC  6396         $DS_BGNSUB
                                      1BBC       T10_S12::
     00010030 9F    000000A8'EF   FA  1BBC                       CALLG   $S$, @#DS$BGNSUB
                                      1BC7  6397         $DS_SETVEC_S    VECTOR=#SCB$L_CSTD,SRVADR=WRITE_SERVICE
                             00   DD  1BC7                       PUSHL   #0
                    000021D4'EF   DF  1BC9                       PUSHAL  WRITE_SERVICE
                    000000F4 8F   DD  1BCF                       PUSHL   #SCB$L_CSTD
              00010160 9F    03   FB  1BD5                       CALLS   #3, @#DS$SETVEC
                    00000000'EF   94  1BDC  6398         CLRB    B_FLAGS                  ; initialize flags
                    00000002'EF   94  1BE2  6399         CLRB    B_FLAGS3
                                      1BE8  6400                                          ; clear buffers
              58    00000000'FF   9E  1BE8  6401         MOVAB   @A_BUFFER_BEGIN,R8       ; load buffer starting address
              59    00000400 8F   D0  1BEF  6402         MOVL    #1024,R9                 ; do two pages
                    00002134'EF   16  1BF6  6403         JSB     CLEAR_BUFFER             ; go clear buffers
                                      1BFC  6404                                          ; send write command
              54    FFE17B80 8F   D0  1BFC  6405         MOVL    #DELAY_TIMES,R4          ; load send packet delay time
              55    00000086'EF   9E  1C03  6406         MOVAB   B_WRITE_3_CMD,R5         ; point to write command packet
                        57   0E   D0  1C0A  6407         MOVL    #14,R7                   ; load transfer byte count
                    0000203D'EF   16  1C0D  6408         JSB     SEND_TU58                ; send command packet
           24 00000000'EF    00   E1  1C13  6409         BBC     #V_CLOCK_FLAG,B_FLAGS,20$ ; br = no timeout error
                                      1C1B  6410         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_INT_ERR,PRLINK=NO_READY
                    00001BEB'EF   DF  1C1B                       PUSHAL  NO_READY
                    000019D1'EF   DF  1C21                       PUSHAL  T_WR_INT_ERR
                             00   DD  1C27                       PUSHL   #0
                             01   DD  1C29                       PUSHL   #$ER
                                      1C2B       ;;; TEST 10, SUBTEST 12, ERROR 1
              000100D0 9F    04   FB  1C2B                       CALLS   #$$M, @#DS$ERRHARD
                                      1C32  6411         $DS_BNCOMPLETE  10$
                          03 50   E9  1C32                       BLBC    R0, 10$
                                      1C35  6412         $DS_ABORT       TEST
                             50   D4  1C35                       CLRL    R0               ; SEND A WARNING TO THE SUPERVISOR
                                  04  1C37                       RET                      ; TERMINATE TEST
                                  1C38  6413 10$:    $DS_ABORT
           00010020 9F     6C    FA  1C38                       CALLG   (AP), @#DS$ABORT
                                  1C3F  6414                                                    ; read continue
           54    FFF3CB00 8F     D0  1C3F  6415 20$:    MOVL     #DELAY_TIME3,R4              ; load receive packet delay time
           55     00000000'FF    9E  1C46  6416         MOVAB    @A_BUFFER_BEGIN,R5           ; load buffer address to store data
               57    55    01    C1  1C4D  6417         ADDL3    #1,R5,R7                     ; load last expected address of data
                  00002083'EF    16  1C51  6418         JSB      RECEIVE_TU58                 ; go get data from TU58
           5A     00000023'EF    9E  1C57  6419         MOVAB    B_CONTINUE,R10               ; load address of expected data
           5B     00000000'FF    9E  1C5E  6420         MOVAB    @A_BUFFER_BEGIN,R11          ; load address of actual data
                     57    55    D1  1C65  6421         CMPL     R5,R7                        ; check for correct number of bytes
                           2A    13  1C68  6422         BEQL     60$                          ; br = correct number of bytes
                                  1C6A  6423 25$:    $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_INT_ERR,PRLINK=REC_ERR,-
                                  1C6A  6424                           P1=#T_REC_CONT
              00001D47'8F    DD  1C6A                       PUSHL   #T_REC_CONT
              00001EB4'EF    DF  1C70                       PUSHAL  REC_ERR
              000019D1'EF    DF  1C76                       PUSHAL  T_WR_INT_ERR
                           00    DD  1C7C                       PUSHL   #0
                           02    DD  1C7E                       PUSHL   #$ER
                                  1C80        ;;; TEST 10, SUBTEST 12, ERROR 2
           000100D0 9F     05    FB  1C80                       CALLS   #$$M, @#DS$ERRHARD
                                  1C87  6425         $DS_BNCOMPLETE  30$
                     03 50    E9  1C87                       BLBC    R0, 30$
                                  1C8A  6426         $DS_ABORT       TEST
                           50    D4  1C8A                       CLRL    R0                    ; SEND A WARNING TO THE SUPERVISOR
                           04        1C8C                       RET                           ; TERMINATE TEST
                                  1C8D  6427
                                  1C8D  6428 30$:    $DS_ABORT
           00010020 9F     6C    FA  1C8D                       CALLG   (AP), @#DS$ABORT
                                  1C94  6429                                                    ; check continue
                     5C    01    D0  1C94  6430 60$:    MOVL     #1,R12                       ; load number of bytes to check
                                  1C97  6431         $DS_BREAK
           00010058 9F     6E    FA  1C97                       CALLG   (SP), @#DS$BREAK
              0000213B'EF    16  1C9E  6432         JSB      CHECK_PACKET                 ; go check continue
                                  1CA4  6433         $DS_BNCOMPLETE  80$                  ; br = no error
                     24 50    E9  1CA4                       BLBC    R0, 80$
                                  1CA7  6434         $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_INT_ERR,PRLINK=DATA_ERR_CONT
              00001E4D'EF    DF  1CA7                       PUSHAL  DATA_ERR_CONT
              000019D1'EF    DF  1CAD                       PUSHAL  T_WR_INT_ERR
                           00    DD  1CB3                       PUSHL   #0
                           03    DD  1CB5                       PUSHL   #$ER
                                  1CB7        ;;; TEST 10, SUBTEST 12, ERROR 3
           000100D0 9F     04    FB  1CB7                       CALLS   #$$M, @#DS$ERRHARD
                                  1CBE  6435         $DS_BNCOMPLETE  70$
                     03 50    E9  1CBE                       BLBC    R0, 70$
                                  1CC1  6436         $DS_ABORT       TEST
                           50    D4  1CC1                       CLRL    R0                    ; SEND A WARNING TO THE SUPERVISOR
                           04        1CC3                       RET                           ; TERMINATE TEST
                                  1CC4  6437 70$:    $DS_ABORT
           00010020 9F     6C    FA  1CC4                       CALLG   (AP), @#DS$ABORT
                                  1CCB  6438                                                    ; send data packet
              00000000'EF    94  1CCB  6439 80$:    CLRB     B_FLAGS                      ; clear byte flags
              00000001'EF    94  1CD1  6440         CLRB     B_FLAGS2                     ; clear byte flags
              00000002'EF    94  1CD7  6441         CLRB     B_FLAGS3                     ; clear byte flags
           55     0000070A'EF    9E  1CDD  6442         MOVAB    B_DATA_PKT_2,R5              ; point to data packet
     57    55     00000084 8F    C1  1CE4  6443         ADDL3    #132,R5,R7                   ; load pointer to end of data
```

```
          19   FF7E4CE0 8F    DA  1CEC  6444          MTPR    #DELAY_TIME7,#PR750$_NICR        ; load interval register 5.2
        00000000'8F    10    DA  1CF3  6445          MTPR    #M_TRANSFER,#PR$_ICCS    ; transfer time to ICS
                 19    01    DA  1CFA  6446          MTPR    #1,#PR750$_NICR          ; load very long next interval   5.2
   00000000'8F   80000041 8F DA  1CFD  6447          MTPR    #<M_ERROR+M_ENABLE+M_RUN>,#PR$_ICCS ; start timer
          1E   00000040 8F   DA  1D08  6448          MTPR    #M_ENABLE,#PR750$_CSTS   ; enable send interrupt          5.2
                                 1D0F  6449
                                 1D0F  6450 85$:     $DS_BREAK
        00010058 9F    6E    FA  1D0F                        CALLG   (SP), @#DS$BREAK
     08 00000002'EF    01    E0  1D16  6451          BBS     #V_SDONE_FLAG,B_FLAGS3,87$ ; br = finished sending
     E9 00000000'EF    00    E1  1D1E  6452          BBC     #V_CLOCK_FLAG,B_FLAGS,85$ ; br = no timeout
                 1E    00    DA  1D26  6453 87$:     MTPR    #0,#PR750$_CSTS          ; disable send interrupt         5.2
                 57    55    D1  1D29  6454          CMPL    R5,R7                    ; check number of bytes sent
                       24    13  1D2C  6455          BEQL    100$                     ; br = correct number of bytes sent
                                 1D2E  6456          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_INT_ERR,PRLINK=SEND_INT_ERR
          00001AC9'EF    DF  1D2E                        PUSHAL  SEND_INT_ERR
          000019D1'EF    DF  1D34                        PUSHAL  T_WR_INT_ERR
                       00    DD  1D3A                        PUSHL   #0
                       04    DD  1D3C                        PUSHL   #$ER
                                 1D3E            ;;; TEST 10, SUBTEST 12, ERROR 4
        000100D0 9F    04    FB  1D3E                        CALLS   #$$M, @#DS$ERRHARD
                                 1D45  6457          $DS_BNCOMPLETE  90$
               03 50    E9  1D45                        BLBC    R0, 90$
                                 1D48  6458          $DS_ABORT       TEST
                       50    D4  1D48                        CLRL    R0                 ; SEND A WARNING TO THE SUPERVISOR
                             04  1D4A                        RET                        ; TERMINATE TEST
                                 1D4B  6459 90$:     $DS_ABORT
        00010020 9F    6C    FA  1D4B                        CALLG   (AP), @#DS$ABORT
                                 1D52  6460                                           ; read end packet
          00000002'EF    96  1D52  6461 100$:    INCB    B_FLAGS3                 ; set data done flag
        54   FF7E4CE0 8F    D0  1D58  6462          MOVL    #DELAY_TIME7,R4          ; load receive packet delay time
        55   00000000'FF    9E  1D5F  6463          MOVAB   @A_BUFFER_BEGIN,R5       ; load address to store actual data
           57   55    0E    C1  1D66  6464          ADDL3   #14,R5,R7                ; load last expected address of data
          00002083'EF    16  1D6A  6465          JSB     RECEIVE_TUSB             ; go get data from TUSB
        5A   0000060C'EF    9E  1D70  6466          MOVAB   B_END_PACKET_S,R10       ; load address of expected end packet
        5B   00000000'FF    9E  1D77  6467          MOVAB   @A_BUFFER_BEGIN,R11      ; load address of actual end packet
                 57    55    D1  1D7E  6468          CMPL    R5,R7                    ; check for correct number of bytes
                       31    13  1D81  6469          BEQL    140$                     ; br = correct number of bytes
        00000000'EF    08    88  1D83  6470          BISB2   #M_END_FLAG,B_FLAGS      ; set end error flag
                                 1D8A  6471 105$:    $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_INT_ERR,PRLINK=REC_ERR,-
                                 1D8A  6472                          P1=#T_REC_END
          00001D2E'8F    DD  1D8A                        PUSHL   #T_REC_END
          00001EB4'EF    DF  1D90                        PUSHAL  REC_ERR
          000019D1'EF    DF  1D96                        PUSHAL  T_WR_INT_ERR
                       00    DD  1D9C                        PUSHL   #0
                       05    DD  1D9E                        PUSHL   #$ER
                                 1DA0            ;;; TEST 10, SUBTEST 12, ERROR 5
        000100D0 9F    05    FB  1DA0                        CALLS   #$$M, @#DS$ERRHARD
                                 1DA7  6473          $DS_BNCOMPLETE  110$
               03 50    E9  1DA7                        BLBC    R0, 110$
                                 1DAA  6474          $DS_ABORT       TEST
                       50    D4  1DAA                        CLRL    R0                 ; SEND A WARNING TO THE SUPERVISOR
                             04  1DAC                        RET                        ; TERMINATE TEST
                                 1DAD  6475
                                 1DAD  6476 110$:    $DS_ABORT
        00010020 9F    6C    FA  1DAD                        CALLG   (AP), @#DS$ABORT
```

```
                                      1DB4  6477                                         ; check end packet
                        5C    0E  D0  1DB4  6478 140$:   MOVL    #14,R12                 ; load number of bytes to check
                                      1DB7  6479         $DS_BREAK
                00010058 9F   6E  FA  1DB7                       CALLG   (SP), @#DS$BREAK
                     0000213B'EF  16  1DBE  6480         JSB     CHECK_PACKET            ; go check end packet
                                      1DC4  6481         $DS_BNCOMPLETE  160$            ; br = no error
                           24 50  E9  1DC4                       BLBC    R0, 160$
                                      1DC7  6482         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_INT_ERR,PRLINK=DATA_ERR_END
                     00001E9D'EF  DF  1DC7                       PUSHAL  DATA_ERR_END
                     000019D1'EF  DF  1DCD                       PUSHAL  T_WR_INT_ERR
                              00  DD  1DD3                       PUSHL   #0
                              06  DD  1DD5                       PUSHL   #$ER
                                      1DD7       ;;; TEST 10, SUBTEST 12, ERROR 6
                000100D0 9F   04  FB  1DD7                       CALLS   #$$M, @#DS$ERRHARD
                                      1DDE  6483         $DS_BNCOMPLETE  150$
                           03 50  E9  1DDE                       BLBC    R0, 150$
                                      1DE1  6484         $DS_ABORT       TEST
                              50  D4  1DE1                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                  04  1DE3                       RET                     ; TERMINATE TEST
                                      1DE4  6485 150$:   $DS_ABORT
                00010020 9F   6C  FA  1DE4                       CALLG   (AP), @#DS$ABORT
                                      1DEB  6486 160$:
                                      1DEB  6487         $DS_ENDSUB
                                      1DEB       T10_S12_X:
        00010038 9F  000000A8'EF  FA  1DEB                       CALLG   $S$, @#DS$ENDSUB
                                      1DF6  6488         $DS_PAGE
```

```
1DF6             .SBTTL  SUBTEST 10.13:   READ RECORD INTERRUPT
1DF6       ;+
1DF6       ; SUBTEST DESCRIPTION:
1DF6       ;
1DF6       ;       This subtest sends the TU58 a command packet to read the last 128
1DF6       ;       bytes of data that were written in the previous subtest in the inter-
1DF6       ;       rupt-driven mode. A timer is set which should allow sufficient time
1DF6       ;       to read a complete record and end packet. An error is reported if
1DF6       ;       this does not occur within the allotted time. The end packet and data
1DF6       ;       packet are checked for errors. If any errors are encountered they are
1DF6       ;       reported.
1DF6       ;
1DF6       ;
1DF6       ; SUBTEST STEPS:
1DF6       ;
1DF6       ;       SUBTEST READ RECORD INTERRUPT
1DF6       ;       :  Clear buffer
1DF6       ;       :  Point to read command packet
1DF6       ;       :  REPEAT
1DF6       ;       :  :  Send command packet byte at a time
1DF6       ;       :  :  Set timer
1DF6       ;       :  :  IF timer expired and no response
1DF6       ;       :  :  :  THEN
1DF6       ;       :  :  :  Report no response error
1DF6       ;       :  :  :  Exit test
1DF6       ;       :  :  ENDIF
1DF6       ;       :  : UNTIL all bytes sent
1DF6       ;       :  ENDREPEAT
1DF6       ;       :  ENABLE read interrupt
1DF6       ;       :  SET timer
1DF6       ;       :  REPEAT
1DF6       ;       :  :  IF interrupt occurs
1DF6       ;       :  :  :  THEN
1DF6       ;       :  :  :  :  Read byte
1DF6       ;       :  :  ENDIF
1DF6       ;       :  : UNTIL timer expired
1DF6       ;       :  ENDREPEAT
1DF6       ;       :  IF expected byte count NEQ actual byte count
1DF6       ;       :  :  THEN
1DF6       ;       :  :  :  DISABLE read interrupt
1DF6       ;       :  :  :  Report error
1DF6       ;       :  :  :  Exit test
1DF6       ;       :  ENDIF
1DF6       ;       :  DISABLE read interrupt
1DF6       ;       :  Point to expected data packet
1DF6       ;       :  REPEAT
1DF6       ;       :  :  IF data expected NEQ data received
1DF6       ;       :  :  :  THEN
1DF6       ;       :  :  :  Report data error
1DF6       ;       :  :  :  Exit subtest
1DF6       ;       :  :  ENDIF
1DF6       ;       :  :  Point to next byte of data
1DF6       ;       :  : UNTIL all data in packet checked
1DF6       ;       :  ENDREPEAT
1DF6       ;       :  Point to expected end packet
```

```
                                     1DF6  6545 ;          :  REPEAT
                                     1DF6  6546 ;          :  :  IF expected end packet data NEQ received end packet data
                                     1DF6  6547 ;          :  :  :  THEN
                                     1DF6  6548 ;          :  :  :  Report end packet data error
                                     1DF6  6549 ;          :  :  :  Exit subtest
                                     1DF6  6550 ;          :  :  ENDIF
                                     1DF6  6551 ;          :  : UNTIL all end packet bytes checked
                                     1DF6  6552 ;          :  ENDREPEAT
                                     1DF6  6553 ;          ENDSUBTEST READ RECORD INTERRUPT
                                     1DF6  6554 ;
                                     1DF6  6555 ;
                                     1DF6  6556 ;
                                     1DF6  6557 ;.page
                                     1DF6  6558 ;-
                                     1DF6  6559            $DS_BGNSUB
                                     1DF6       T10_S13::
    00010030 9F   000000B4'EF   FA   1DF6                       CALLG    $S$, @#DS$BGNSUB
                                     1E01  6560            $DS_SETVEC_S      VECTOR=#SCB$L_CSRD,SRVADR=READ_SERVICE
                            00  DD   1E01                       PUSHL    #0
                  00002164'EF   DF   1E03                       PUSHAL   READ_SERVICE
                  000000F0 8F   DD   1E09                       PUSHL    #SCB$L_CSRD
         00010160 9F        03  FB   1E0F                       CALLS    #3, @#DS$SETVEC
                  00000000'EF   94   1E16  6561            CLRB     B_FLAGS                   ; clear byte flags
                  00000001'EF   94   1E1C  6562            CLRB     B_FLAGS2                  ; clear byte flags
         58       00000000'FF   9E   1E22  6563            MOVAB    @A_BUFFER_BEGIN,R8        ; load buffer start address
         59       00000400 8F   D0   1E29  6564            MOVL     #1024,R9                  ; load number of bytes to clear
                  00002134'EF   16   1E30  6565            JSB      CLEAR_BUFFER              ; go clear buffer
                        56  1D  DB   1E36  6566            MFPR     #PR750$_CSRD,R6           ; dump data buffer to clear done   5.2
         1C       00000040 8F   DA   1E39  6567            MTPR     #M_ENABLE,#PR750$_CSRS    ; enable read interrupt            5.2
         53       00000000'FF   9E   1E40  6568            MOVAB    @A_BUFFER_BEGIN,R3        ; load buffer address
      59 00000092 8F        53  C1   1E47  6569            ADDL3    R3,#146,R9                ; load expected end of buffer
         55       00000094'EF   9E   1E4F  6570            MOVAB    B_READ_3_CMD,R5           ; point to read record command packet
                        57  0E  D0   1E56  6571            MOVL     #14,R7                    ; load byte count
                  0000203D'EF   16   1E59  6572            JSB      SEND_TU58                 ; send command packet
        24 00000000'EF      00  E1   1E5F  6573            BBC      #V_CLOCK_FLAG,B_FLAGS,20$ ; br = no timeout error
                                     1E67  6574            $DS_ERRHARD_S      ,UNIT=#0,MSGADR=T_RD_INT_ERR,PRLINK=NO_READY
                  00001BEB'EF   DF   1E67                       PUSHAL   NO_READY
                  0000199F'EF   DF   1E6D                       PUSHAL   T_RD_INT_ERR
                            00  DD   1E73                       PUSHL    #0
                            01  DD   1E75                       PUSHL    #$ER
                                     1E77       ;;; TEST 10, SUBTEST 13, ERROR 1
         000100D0 9F        04  FB   1E77                       CALLS    #$$M, @#DS$ERRHARD
                                     1E7E  6575            $DS_BNCOMPLETE  10$
                         03 50  E9   1E7E                       BLBC     R0, 10$
                                     1E81  6576            $DS_ABORT         TEST
                            50  D4   1E81                       CLRL     R0                   ; SEND A WARNING TO THE SUPERVISOR
                            04       1E83                       RET                           ; TERMINATE TEST
                                     1E84  6577 10$:       $DS_ABORT
         00010020 9F        6C  FA   1E84                       CALLG    (AP), @#DS$ABORT
                  00000000'EF   94   1E8B  6578 20$:       CLRB     B_FLAGS                   ; clear byte flags
         19       FF7E4CE0 8F   DA   1E91  6579            MTPR     #DELAY_TIME7,#PR750$_NICR       ; load timer interval   5.2
         00000000'8F        10  DA   1E98  6580            MTPR     #M_TRANSFER,#PR$_ICCS     ; transfer time to ICS
                        19  01  DA   1E9F  6581            MTPR     #1,#PR750$_NICR           ; load very long next interval  5.2
   00000000'8F    80000041 8F   DA   1EA2  6582            MTPR     #<M_ERROR+M_ENABLE+M_RUN>,#PR$_ICCS ; start timer
                                     1EAD  6583 30$:       $DS_BREAK
        00010058 9F    6E    FA  1EAD                       CALLG   (SP), @#DS$BREAK
   F1 00000000'EF      00    E1  1EB4  6584         BBC     #V_CLOCK_FLAG,B_FLAGS,30$ ; br = no timeout
                1C     00    DA  1EBC  6585         MTPR    #0,#PR750$_CSRS           ; disable read interrupt          5.2
      5B   00000000'FF       9E  1EBF  6586         MOVAB   @A_BUFFER_BEGIN,R11       ; load address of actual data
      5A   0000070A'EF       9E  1EC6  6587         MOVAB   B_DATA_PKT_2,R10          ; load address of expected data
                59     53    D1  1ECD  6588         CMPL    R3,R9                     ; correct number of bytes received ?
                       34    13  1ED0  6589         BEQL    50$                       ; br = correct number of bytes received
        00000000'EF    04    88  1ED2  6590         BISB2   #M_DATA_FLAG,B_FLAGS      ; set data error flag
                55     53    D0  1ED9  6591         MOVL    R3,R5                     ; point to last address loaded
                                 1EDC  6592         $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_RD_INT_ERR,PRLINK=REC_ERR,-
                                 1EDC  6593                           P1=#T_REC_DATA
           00001D68'8F       DD  1EDC                       PUSHL   #T_REC_DATA
           00001EB4'EF       DF  1EE2                       PUSHAL  REC_ERR
           0000199F'EF       DF  1EE8                       PUSHAL  T_RD_INT_ERR
                       00    DD  1EEE                       PUSHL   #0
                       02    DD  1EF0                       PUSHL   #$ER
                                 1EF2           ;;; TEST 10, SUBTEST 13, ERROR 2
        000100D0 9F    05    FB  1EF2                       CALLS   #$$M, @#DS$ERRHARD
                                 1EF9  6594         $DS_BNCOMPLETE  40$
                    03 50    E9  1EF9                       BLBC    R0, 40$
                                 1EFC  6595         $DS_ABORT       TEST
                       50    D4  1EFC                       CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                       04        1EFE                       RET                         ; TERMINATE TEST
                                 1EFF  6596 40$:    $DS_ABORT
        00010020 9F    6C    FA  1EFF                       CALLG   (AP), @#DS$ABORT
      5C   00000084 8F       D0  1F06  6597 50$:    MOVL    #132,R12                  ; load number of bytes to check
                                 1F0D  6598         $DS_BREAK
        00010058 9F    6E    FA  1F0D                       CALLG   (SP), @#DS$BREAK
           0000213B'EF       16  1F14  6599         JSB     CHECK_PACKET              ; go check data packet
                                 1F1A  6600         $DS_BNCOMPLETE  70$               ; br = no data error
                    24 50    E9  1F1A                       BLBC    R0, 70$
                                 1F1D  6601         $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_RD_INT_ERR,PRLINK=DATA_ERR_DATA
           00001E75'EF       DF  1F1D                       PUSHAL  DATA_ERR_DATA
           0000199F'EF       DF  1F23                       PUSHAL  T_RD_INT_ERR
                       00    DD  1F29                       PUSHL   #0
                       03    DD  1F2B                       PUSHL   #$ER
                                 1F2D           ;;; TEST 10, SUBTEST 13, ERROR 3
        000100D0 9F    04    FB  1F2D                       CALLS   #$$M, @#DS$ERRHARD
                                 1F34  6602         $DS_BNCOMPLETE  60$
                    03 50    E9  1F34                       BLBC    R0, 60$
                                 1F37  6603         $DS_ABORT       TEST
                       50    D4  1F37                       CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                       04        1F39                       RET                         ; TERMINATE TEST
                                 1F3A  6604 60$:    $DS_ABORT
        00010020 9F    6C    FA  1F3A                       CALLG   (AP), @#DS$ABORT
      5A   0000060C'EF       9E  1F41  6605 70$:    MOVAB   B_END_PACKET_S,R10        ; load address of expected end packet
      5B   00000084 8F       C0  1F48  6606         ADDL2   #132,R11                  ; adjust pointer to actual end packet
                5C     0E    D0  1F4F  6607         MOVL    #14,R12                   ; load number of bytes to check
                                 1F52  6608         $DS_BREAK
        00010058 9F    6E    FA  1F52                       CALLG   (SP), @#DS$BREAK
           0000213B'EF       16  1F59  6609         JSB     CHECK_PACKET              ; go check end packet
                                 1F5F  6610         $DS_BNCOMPLETE  90$               ; br = no data error
                    24 50    E9  1F5F                       BLBC    R0, 90$
                                 1F62  6611         $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_RD_INT_ERR,PRLINK=DATA_ERR END
           00001E9D'EF       DF  1F62                       PUSHAL  DATA_ERR_END
```

```
               0000199F'EF    DF  1F68                          PUSHAL  T_RD_INT_ERR
                        00    DD  1F6E                          PUSHL   #0
                        04    DD  1F70                          PUSHL   #$ER
                                  1F72       ;;; TEST 10, SUBTEST 13, ERROR 4
          000100D0 9F   04    FB  1F72                          CALLS   #$$M, @#DS$ERRHARD
                                  1F79  6612          $DS_BNCOMPLETE 80$
                     03 50    E9  1F79                          BLBC    R0, 80$
                                  1F7C  6613          $DS_ABORT       TEST
                        50    D4  1F7C                          CLRL    R0                 ; SEND A WARNING TO THE SUPERVISOR
                        04        1F7E                          RET                        ; TERMINATE TEST
                                  1F7F  6614 80$:     $DS_ABORT
          00010020 9F   6C    FA  1F7F                          CALLG   (AP), @#DS$ABORT
                                  1F86  6615 90$:
                                  1F86  6616          $DS_ENDSUB
                                  1F86       T10_S13_X:
 00010038 9F   000000B4'EF    FA  1F86                          CALLG   $S$, @#DS$ENDSUB
                                  1F91  6617          $DS_PAGE
```

```
1F91                .SBTTL  SUBTEST 10.14:  IPL CHECK
1F91          ;+
1F91          ; SUBTEST DESCRIPTION:
1F91          ;
1F91          ;       This subtest checks that the TU58 interrupt occurs at the cor-
1F91          ;       rect interrupt priority level.  The IPL is set one level above
1F91          ;       that at which a TU58 interrupt should occur.  A NOP command is
1F91          ;       issued to cause the return  of an end packet in interrupt mode.
1F91          ;       If an interrupt occurs an error is reported.  The IPL is then
1F91          ;       decremented and if an interrupt does not occur within a reason-
1F91          ;       interval an error is reported.  There are no data checks made.
1F91          ;
1F91          ;
1F91          ; SUBTEST STEPS:
1F91          ;
1F91          ;       SUBTEST IPL CHECK
1F91          ;       :  Point to NOP command packet
1F91          ;       :  REPEAT
1F91          ;       :  :  SEND command packet byte at a time
1F91          ;       :  :  SET timer
1F91          ;       :  :  IF timer expired AND no response
1F91          ;       :  :  :  THEN
1F91          ;       :  :  :  :  REPORT no response error
1F91          ;       :  :  :  :  Exit test
1F91          ;       :  :  ENDIF
1F91          ;       :  : UNTIL all bytes sent
1F91          ;       :  ENDREPEAT
1F91          ;       :  SET IPL above read interrupt
1F91          ;       :  SET timer
1F91          ;       :  ENABLE read interrupt
1F91          ;       :  IF interrupt occurs before timeout
1F91          ;       :  :  THEN
1F91          ;       :  :  :  DISABLE read interrupt
1F91          ;       :  :  :  REPORT error
1F91          ;       :  :  :  Abort test
1F91          ;       :  :  ENDIF
1F91          ;       :  DISABLE read interrupt
1F91          ;       :  DECR IPL
1F91          ;       :  SET timer
1F91          ;       :  ENABLE read interrupt
1F91          ;       :  IF timer expires AND no interrupt
1F91          ;       :  :  THEN
1F91          ;       :  :  :  DISABLE read interrupt
1F91          ;       :  :  :  REPORT error
1F91          ;       :  :  :  Abort test
1F91          ;       :  ENDIF
1F91          ;       :  DISABLE read interrupt
1F91          ;       :  CLEAR IPL
1F91          ;       ENDSUBTEST IPL CHECK
1F91          ;
1F91          ;
1F91          ;
1F91          ;.page
1F91          ;-
1F91
```

```
                                  1F91  6674          $DS_BGNSUB
                                  1F91           T10_S14::
    00010030 9F    000000C0'EF    FA  1F91                  CALLG   $S$, @#DS$BGNSUB
                                  1F9C  6675          $DS_CLRVEC_S    VECTOR=#SCB$L_CSRD
          000000F0 8F    DD  1F9C                   PUSHL   #SCB$L_CSRD
       00010168 9F    01    FB  1FA2                   CALLS   #1, @#DS$CLRVEC
                                  1FA9  6676          $DS_SETVEC_S    VECTOR=#SCB$L_CSRD,SRVADR=READ_SERVICE2
                    00    DD  1FA9                   PUSHL   #0
          000021A0'EF    DF  1FAB                   PUSHAL  READ_SERVICE2
          000000F0 8F    DD  1FB1                   PUSHL   #SCB$L_CSRD
       00010160 9F    03    FB  1FB7                   CALLS   #3, @#DS$SETVEC
          00000000'EF    94  1FBE  6677          CLRB    B_FLAGS                   ; clear byte flags
          00000001'EF    94  1FC4  6678          CLRB    B_FLAGS2                  ; clear byte flags
          00000002'EF    94  1FCA  6679          CLRB    B_FLAGS3                  ; clear byte flags
       54    FFFB6C20 8F    D0  1FD0  6680          MOVL    #DELAY_TIME1,R4           ; load send packet delay time
       55    00000032'EF    9E  1FD7  6681          MOVAB   B_NOP_CMD,R5              ; point to init command packet
                 57    0E    D0  1FDE  6682          MOVL    #14,R7                    ; load transfer byte count
          0000203D'EF    16  1FE1  6683          JSB     SEND_TU58                 ; send command packet
     24 00000000'EF    00    E1  1FE7  6684          BBC     #V_CLOCK_FLAG,B_FLAGS,20$ ; br = no timeout error
                                  1FEF  6685          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_IPL_ERR,PRLINK=NO_READY
          00001BEB'EF    DF  1FEF                   PUSHAL  NO_READY
          00001A04'EF    DF  1FF5                   PUSHAL  T_IPL_ERR
                    00    DD  1FFB                   PUSHL   #0
                    01    DD  1FFD                   PUSHL   #$ER
                                  1FFF           ;;; TEST 10, SUBTEST 14, ERROR 1
       000100D0 9F    04    FB  1FFF                   CALLS   #$$M, @#DS$ERRHARD
                                  2006  6686          $DS_BNCOMPLETE  10$
                 03 50    E9  2006                   BLBC    R0, 10$
                                  2009  6687          $DS_ABORT       TEST
                    50    D4  2009                   CLRL    R0                ; SEND A WARNING TO THE SUPERVISOR
                    04       200B                   RET                       ; TERMINATE TEST
                                  200C  6688 10$:     $DS_ABORT
       00010020 9F    6C    FA  200C                   CALLG   (AP), @#DS$ABORT
                                  2013  6689
          00000000'EF    94  2013  6690 20$:     CLRB    B_FLAGS                   ; clear byte flags
                 56    1D    DB  2019  6691          MFPR    #PR750$_CSRD,R6           ; dump buffer to clear done      5.2
                                  201C  6692          $DS_SETIPL_S    LEVEL=#^X17       ; set IPL above read interrupt
                    17    DD  201C                   PUSHL   #^X17
       00010178 9F    01    FB  201E                   CALLS   #1, @#DS$SETIPL
       19    FFF3CB00 8F    DA  2025  6693          MTPR    #DELAY_TIME3,#PR750$_NICR       ; load timer interval   5.2
       00000000'8F    10    DA  202C  6694          MTPR    #M_TRANSFER,#PR$_ICCS     ; transfer time to ICS
                 19    01    DA  2033  6695          MTPR    #1,#PR750$_NICR           ; load very long next interval  5.2
    00000000'8F    80000041 8F    DA  2036  6696          MTPR    #<M_ERROR+M_ENABLE+M_RUN>,#PR$_ICCS ; start timer
       1C    00000040 8F    DA  2041  6697          MTPR    #M_ENABLE,#PR750$_CSRS    ; enable read interrupt          5.2
                                  2048  6698 30$:     $DS_BREAK                         ; enable break
       00010058 9F    6E    FA  2048                   CALLG   (SP), @#DS$BREAK
     F1 00000000'EF    00    E1  204F  6699          BBC     #V_CLOCK_FLAG,B_FLAGS,30$ ; br = no timeout
     27 00000002'EF    02    E1  2057  6700          BBC     #V_RD_INT_FLAG,B_FLAGS3,50$ ; br = no interrupt
                 1C    00    DA  205F  6701          MTPR    #0,#PR750$_CSRS           ; disable read interrupt         5.2
                                  2062  6702          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_IPL_ERR,PRLINK=INT_ERR
          00001F3D'EF    DF  2062                   PUSHAL  INT_ERR
          00001A04'EF    DF  2068                   PUSHAL  T_IPL_ERR
                    00    DD  206E                   PUSHL   #0
                    02    DD  2070                   PUSHL   #$ER
                                  2072           ;;; TEST 10, SUBTEST 14, ERROR 2
```

```
              000100D0 9F    04   FB  2072                          CALLS   #$$M, @#DS$ERRHARD
                                     2079  6703         $DS_BNCOMPLETE  40$
                         03 50   E9  2079                          BLBC    R0, 40$
                                     207C  6704         $DS_ABORT       TEST
                            50   D4  207C                          CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  207E                          RET                     ; TERMINATE TEST
                                     207F  6705 40$:    $DS_ABORT
              00010020 9F    6C   FA  207F                          CALLG   (AP), @#DS$ABORT
                                     2086  6706
                       1C    00   DA  2086  6707 50$:    MTPR    #0,#PR750$_CSRS         ; disable read interrupt        5.2
                   00000002'EF    94  2089  6708         CLRB    B_FLAGS3                ; clear byte flags
              19   FF7E4CE0 8F    DA  208F  6709         MTPR    #DELAY_TIME7,#PR750$_NICR   ; load timer interval       5.2
              00000000'8F    10   DA  2096  6710         MTPR    #M_TRANSFER,#PR$_ICCS   ; transfer time to ICS
                       19    01   DA  209D  6711         MTPR    #1,#PR750$_NICR         ; load very long next interval  5.2
     00000000'8F   80000041 8F    DA  20A0  6712         MTPR    #<M_ERROR+M_ENABLE+M_RUN>,#PR$_ICCS ; start timer
              1C   00000040 8F    DA  20AB  6713         MTPR    #M_ENABLE,#PR750$_CSRS  ; enable read interrupt         5.2
                                     20B2  6714         $DS_SETIPL_S    LEVEL=#^X16     ; set IPL to allow read interrupt
                            16   DD  20B2                          PUSHL   #^X16
              00010178 9F    01   FB  20B4                          CALLS   #1, @#DS$SETIPL
                                     20BB  6715 60$:    $DS_BREAK                       ; enable break
              00010058 9F    6E   FA  20BB                          CALLG   (SP), @#DS$BREAK
           2F 00000002'EF    02   E0  20C2  6716         BBS     #V_RD_INT_FLAG,B_FLAGS3,80$ ; br = interrupt
           E9 00000000'EF    00   E1  20CA  6717         BBC     #V_CLOCK_FLAG,B_FLAGS,60$ ; br = no timeout
                       1C    00   DA  20D2  6718         MTPR    #0,#PR750$_CSRS         ; disable read interrupt        5.2
                                     20D5  6719         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_IPL_ERR,PRLINK=NO_INT_ERR
                   00001F4B'EF    DF  20D5                          PUSHAL  NO_INT_ERR
                   00001A04'EF    DF  20DB                          PUSHAL  T_IPL_ERR
                            00   DD  20E1                          PUSHL   #0
                            03   DD  20E3                          PUSHL   #$ER
                                     20E5       ;;; TEST 10, SUBTEST 14, ERROR 3
              000100D0 9F    04   FB  20E5                          CALLS   #$$M, @#DS$ERRHARD
                                     20EC  6720         $DS_BNCOMPLETE  70$
                         03 50   E9  20EC                          BLBC    R0, 70$
                                     20EF  6721         $DS_ABORT       TEST
                            50   D4  20EF                          CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  20F1                          RET                     ; TERMINATE TEST
                                     20F2  6722 70$:    $DS_ABORT
              00010020 9F    6C   FA  20F2                          CALLG   (AP), @#DS$ABORT
                                     20F9  6723
                       1C    00   DA  20F9  6724 80$:    MTPR    #0,#PR750$_CSRS         ; disable interrupt             5.2
                                     20FC  6725         $DS_SETIPL_S    LEVEL=#0        ; restore IPL
                            00   DD  20FC                          PUSHL   #0
              00010178 9F    01   FB  20FE                          CALLS   #1, @#DS$SETIPL
                                     2105  6726
                                     2105  6727         $DS_ENDSUB
                                     2105       T10_S14_X:
     00010038 9F   000000C0'EF    FA  2105                          CALLG   $S$, @#DS$ENDSUB
                                     2110  6728
                                     2110  6729         $DS_CLRVEC_S    VECTOR=#SCB$L_TIMER
                   000000C0 8F    DD  2110                          PUSHL   #SCB$L_TIMER
              00010168 9F    01   FB  2116                          CALLS   #1, @#DS$CLRVEC
                                     211D  6730         $DS_CLRVEC_S    VECTOR=#SCB$L_CSRD
                   000000F0 8F    DD  211D                          PUSHL   #SCB$L_CSRD
              00010168 9F    01   FB  2123                          CALLS   #1, @#DS$CLRVEC
                                     212A  6731         $DS_CLRVEC_S    VECTOR-#SCB$L_CSTD
```

```
             000000F4 8F   DD  212A                        PUSHL   #SCB$L_CSTD
         00010168 9F   01   FB  2130                        CALLS   #1, a#DS$CLRVEC
                                2137  6732
                                2137  6733        $DS_ENDTEST
                   50   01   D0  2137                        MOVL    #1, R0          ; NORMAL EXIT
                                213A        TEST_010_X::
         00010058 9F   6E   FA  213A                        CALLG   (SP), a#DS$BREAK
                           04  2141                        RET                     ; RETURN TO TEST SEQUENCER
                                2142  6734
                                2142  6735        $DS_PAGE
```

```
                                     2142  6737
                                     2142              .SBTTL  TEST 11:  TU58 EXERCISER (WITH MRSP)
                                 00000000                      .PSECT  TEST_011, PAGE, NOWRT
                                     0020
                                     0020  6739 ;++
                                     0020  6740 ; Test description:
                                     0020  6741 ;
                                     0020  6742 ;       This test exercises the TU58 on the VAX-11/750 by performing
                                     0020  6743 ;       initialization, self diagnostic and no operation.
                                     0020  6744 ;       All information received from the TU58 is checked for errors.
                                     0020  6745 ;       All errors detected are reported.
                                     0020  6746 ;
                                     0020  6747 ;
                                     0020  6748 ; Assumptions:
                                     0020  6749 ;
                                     0020  6750 ;       This subtest does not write or read the TU58. All operations performed
                                     0020  6751 ;       deal with the controller only. (NOTE: Test 2 requires a scratch tape).
                                     0020  6752 ;
                                     0020  6753 ;
                                     0020  6754 ;.page
                                     0020  6755 ;--
                                     0020  6756
                                     0020  6757         $DS_BGNTEST     <DEFAULT,TU58,ALL>
                                     0020       DATA_011:
                           00000000  0020                      .LONG   0               ; TEST ARGUMENT TABLE TERMINATOR
                                     0024       TEST_011::
                               0000  0024                      .WORD   ^M<>            ; ENTRY MASK
                                     0026  6758
                                     0026  6759
                                     0026  6760         ; Set Up
                                     0026  6761
                                     0026  6762
                                     0026  6763         $DS_SETVEC_S    VECTOR=#SCB$L_TIMER,SRVADR=TIMER_SERVICE
                            00   DD  0026                      PUSHL   #0
                       00002150'EF   DF  0028                      PUSHAL  TIMER_SERVICE
                       000000C0 8F   DD  002E                      PUSHL   #SCB$L_TIMER
                  00010160 9F   03   FB  0034                      CALLS   #3, @#DS$SETVEC
              58   00000000'FF   9E  003B  6764 1$:     MOVAB   @A_BUFFER_BEGIN,R8       ; load buffer starting address
00000013'EF   58   00000092 8F   C1  0042  6765         ADDL3   #146,R8,BUF_LIM          ; construct upper buffer limit
              59   00000400 8F   D0  004E  6766         MOVL    #1024,R9                 ; do both pages
                   00002134'EF   16  0055  6767         JSB     CLEAR_BUFFER             ; go clear buffers
                   00000000'EF   94  005B  6768         CLRB    B_FLAGS                  ; init flags
                   00000001'EF   94  0061  6769         CLRB    B_FLAGS2                 ; init flags
                                     0067  6770
                   00001FCB'EF   16  0067  6771 2$:     JSB     MINIT_SEQUENCE           ; initialize drive
              5A   00000023'EF   9E  006D  6772         MOVAB   B_CONTINUE,R10           ; load expected data address
              5B   00000000'FF   9E  0074  6773         MOVAB   @A_BUFFER_BEGIN,R11      ; load actual data address
           2C 00000000'EF   04   E1  007B  6774         BBC     #V_SEND_FLAG,B_FLAGS,5$ ; br = no send error
           DC 00000001'EF   01   E3  0083  6775         BBCS    #V_REPEAT_SEND,B_FLAGS2,2$ ; br = first try and set flag
                                     008B  6776         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_INIT1_ERR,PRLINK=NO_READY
                   00001BEB'EF   DF  008B                      PUSHAL  NO_READY
                   0000183B'EF   DF  0091                      PUSHAL  T_INIT1_ERR
                            00   DD  0097                      PUSHL   #0
                            01   DD  0099                      PUSHL   #$ER
                                     009B       ;;; TEST 11, SUBTEST 0, ERROR 1
              000100D0 9F   04   FB  009B                       CALLS   #$$M, @#DS$ERRHARD
                                     00A2  6777         $DS_BNCOMPLETE  3$
                         03 50   E9  00A2                       BLBC    R0, 3$
                                     00A5  6778         $DS_ABORT       TEST
                            50   D4  00A5                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  00A7                       RET                     ; TERMINATE TEST
                                     00A8  6779 3$:     $DS_ABORT
              00010020 9F   6C   FA  00A8                       CALLG   (AP), @#DS$ABORT
                                     00AF  6780
           32 00000000'EF   01   E1  00AF  6781 5$:     BBC     #V_RECEIVE_FLAG,B_FLAGS,9$ ; br = no receive error
           A8 00000001'EF   01   E3  00B7  6782         BBCS    #V_REPEAT_SEND,B_FLAGS2,2$ ; br = first try and set flag
                                     00BF  6783 7$:     $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_INIT1_ERR,PRLINK=REC_ERR,-
                                     00BF  6784                         P1=#T_REC_CONT
                   00001D47'8F   DD  00BF                       PUSHL   #T_REC_CONT
                   00001EB4'EF   DF  00C5                       PUSHAL  REC_ERR
                   0000183B'EF   DF  00CB                       PUSHAL  T_INIT1_ERR
                            00   DD  00D1                       PUSHL   #0
                            02   DD  00D3                       PUSHL   #$ER
                                     00D5       ;;; TEST 11, SUBTEST 0, ERROR 2
              000100D0 9F   05   FB  00D5                       CALLS   #$$M, @#DS$ERRHARD
                                     00DC  6785         $DS_BNCOMPLETE  8$
                         03 50   E9  00DC                       BLBC    R0, 8$
                                     00DF  6786         $DS_ABORT       TEST
                            50   D4  00DF                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  00E1                       RET                     ; TERMINATE TEST
                                     00E2  6787 8$:     $DS_ABORT
              00010020 9F   6C   FA  00E2                       CALLG   (AP), @#DS$ABORT
                                     00E9  6788
                       5C   01   D0  00E9  6789 9$:     MOVL    #1,R12                  ; load number of bytes to check
                                     00EC  6790         $DS_BREAK
              00010058 9F   6E   FA  00EC                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16  00F3  6791         JSB     CHECK_PACKET            ; go check continue
                                     00F9  6792         $DS_BNCOMPLETE  12$             ; br = continue received
                         39 50   E9  00F9                       BLBC    R0, 12$
           0A 00000001'EF   01   E0  00FC  6793         BBS     #V_REPEAT_SEND,B_FLAGS2,10$ ; br = first try
              00000001'EF   02   88  0104  6794         BISB2   #M_REPEAT_SEND,B_FLAGS2 ; set repeat flag
                          FF59   31  010B  6795         BRW     2$                      ; try again
                       52   50   D0  010E  6796 10$:    MOVL    R0,R2                   ; save code for error report
                                     0111  6797         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_INIT1_ERR,PRLINK=DATA_ERR_CONT
                   00001E4D'EF   DF  0111                       PUSHAL  DATA_ERR_CONT
                   0000183B'EF   DF  0117                       PUSHAL  T_INIT1_ERR
                            00   DD  011D                       PUSHL   #0
                            03   DD  011F                       PUSHL   #$ER
                                     0121       ;;; TEST 11, SUBTEST 0, ERROR 3
              000100D0 9F   04   FB  0121                       CALLS   #$$M, @#DS$ERRHARD
                                     0128  6798         $DS_BNCOMPLETE  11$
                         03 50   E9  0128                       BLBC    R0, 11$
                                     012B  6799         $DS_ABORT       TEST
                            50   D4  012B                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  012D                       RET                     ; TERMINATE TEST
                                     012E  6800 11$:    $DS_ABORT
              00010020 9F   6C   FA  012E                       CALLG   (AP), @#DS$ABORT
                                     0135  6801 12$:
                                     0135  6802         $DS_PAGE
```

```
0135             .SBTTL  SUBTEST 11.1:    INITIALIZE
0135      ;+
0135      ; SUBTEST DESCRIPTION:
0135      ;
0135      ;       This subtest sends the TU58 a command packet to perform an initialization
0135      ;       operation which puts the controller into a known ready state. While
0135      ;       sending each byte of the packet, a timer is set which will time out if
0135      ;       the ready bit in the transmitter processor register does not set
0135      ;       within the alloted time. The same is done for the done bit in the
0135      ;       receiver processor register while reading a byte from the TU58. When
0135      ;       this is completed successfully, the returned end packet is compared to
0135      ;       an expected end packet. If a time out occurs or bad data is
0135      ;       encountered an error will be reported.
0135      ;
0135      ;
0135      ; SUBTEST STEPS:
0135      ;
0135      ;       SUBTEST INITIALIZE
0135      ;       :  Clear buffer
0135      ;       :  Point to initialize command packet
0135      ;       :  REPEAT
0135      ;       :  :  Send command packet byte at a time
0135      ;       :  :  Set timer
0135      ;       :  :  IF timer expired AND no response
0135      ;       :  :  :  THEN
0135      ;       :  :  :  Report no response error
0135      ;       :  :  :  Exit test
0135      ;       :  :  ENDIF
0135      ;       :  : UNTIL all bytes sent
0135      ;       :  ENDREPEAT
0135      ;       :  REPEAT
0135      ;       :  :  Set timer
0135      ;       :  :  IF timer expired AND done set
0135      ;       :  :  :  THEN
0135      ;       :  :  :  :  Read byte
0135      ;       :  :  ENDIF
0135      ;       :  : UNTIL timer expired AND done clear
0135      ;       :  ENDREPEAT
0135      ;       :  IF expected byte count NEQ actual byte count
0135      ;       :  :  THEN
0135      ;       :  :  :  Report error
0135      ;       :  :  :  Exit test
0135      ;       :  ENDIF
0135      ;       :  Point to expected and actual end packet
0135      ;       :  REPEAT
0135      ;       :  :  IF data expected NEQ data received
0135      ;       :  :  :  THEN
0135      ;       :  :  :  Report error
0135      ;       :  :  :  Exit test
0135      ;       :  :  ENDIF
0135      ;       :  : UNTIL all data checked
0135      ;       :  ENDREPEAT
0135      ;       ENDSUBTEST INITIALIZE
0135      ;
0135      ;
```

```
                                      0135  6859 ;
                                      0135  6860 ;
                                      0135  6861 ;.page
                                      0135  6862 ;-
                                      0135  6863
                                      0135  6864         $DS_BGNSUB
                                      0135       T11_S1::
     00010030 9F   000000CC'EF    FA  0135                       CALLG   $S$, @#DS$BGNSUB
                                      0140  6865
                   00000000'EF    94  0140  6866         CLRB    B_FLAGS                     ; initialize flags
             58    00000000'FF    9E  0146  6867         MOVAB   @A_BUFFER_BEGIN,R8          ; load buffer starting address
             59    00000400 8F    D0  014D  6868         MOVL    #1024,R9                    ; do both pages
                   00002134'EF    16  0154  6869         JSB     CLEAR_BUFFER                ; go clear buffers
             54    FFFB6C20 8F    D0  015A  6870         MOVL    #DELAY_TIME1,R4             ; load send packet delay time
             55    000002FC'EF    9E  0161  6871         MOVAB   MB_INIT_CMD,R5              ; point to init command packet
                         57   0E  D0  0168  6872         MOVL    #14,R7                      ; load transfer byte count
                   0000203D'EF    16  016B  6873         JSB     SEND_TU58                   ; send command packet
          24 00000000'EF     00   E1  0171  6874         BBC     #V_CLOCK_FLAG,B_FLAGS,2$  ; br = no timeout error
                                      0179  6875         $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_INIT2_ERR,PRLINK=NO_READY
                   00001BEB'EF    DF  0179                       PUSHAL  NO_READY
                   00001865'EF    DF  017F                       PUSHAL  T_INIT2_ERR
                             00   DD  0185                       PUSHL   #0
                             01   DD  0187                       PUSHL   #$ER
                                      0189       ;;; TEST 11, SUBTEST 1, ERROR 1
          000100D0 9F        04   FB  0189                       CALLS   #$$M, @#DS$ERRHARD
                                      0190  6876         $DS_BNCOMPLETE  1$
                          03 50   E9  0190                       BLBC    R0, 1$
                                      0193  6877         $DS_ABORT       TEST
                             50   D4  0193                       CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                             04   0195                           RET                         ; TERMINATE TEST
                                      0196  6878 1$:     $DS_ABORT
          00010020 9F        6C   FA  0196                       CALLG   (AP), @#DS$ABORT
                                      019D  6879
          54    FFFB6C20 8F       D0  019D  6880 2$:     MOVL    #DELAY_TIME1,R4             ; load receive packet delay time
          55     00000000'FF      9E  01A4  6881         MOVAB   @A_BUFFER_BEGIN,R5          ; load buffer address to store data
             57     55     0E     C1  01AB  6882         ADDL3   #14,R5,R7                   ; load last expected address of data
                000020C8'EF       16  01AF  6883         JSB     MRECEIVE_TU58               ; go get data from TU58
          5A     00000660'EF      9E  01B5  6884         MOVAB   MB_END_PACKET_1,R10         ; load address of expected end packet
          5B     00000000'FF      9E  01BC  6885         MOVAB   @A_BUFFER_BEGIN,R11         ; load address of actual end packet
                     57      55   D1  01C3  6886         CMPL    R5,R7                       ; check for correct number of bytes
                             31   13  01C6  6887         BEQL    6$                          ; br = correct number of bytes
          00000000'EF        08   88  01C8  6888         BISB2   #M_END_FLAG,B_FLAGS         ; set end error flag
                                      01CF  6889 4$:     $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_INIT2_ERR,PRLINK=REC_ERR,-
                                      01CF  6890                         P1=#T_REC_END
                   00001D2E'8F    DD  01CF                       PUSHL   #T_REC_END
                   00001EB4'EF    DF  01D5                       PUSHAL  REC_ERR
                   00001865'EF    DF  01DB                       PUSHAL  T_INIT2_ERR
                             00   DD  01E1                       PUSHL   #0
                             02   DD  01E3                       PUSHL   #$ER
                                      01E5       ;;; TEST 11, SUBTEST 1, ERROR 2
          000100D0 9F        05   FB  01E5                       CALLS   #$$M, @#DS$ERRHARD
                                      01EC  6891         $DS_BNCOMPLETE  5$
                          03 50   E9  01EC                       BLBC    R0, 5$
                                      01EF  6892         $DS_ABORT       TEST
                             50   D4  01EF                       CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
```

```
                                 04  01F1                       RET                     ; TERMINATE TEST
                                     01F2  6893 5$:     $DS_ABORT
              00010020 9F   6C   FA  01F2                       CALLG   (AP), @#DS$ABORT
                                     01F9  6894
                       5C   0E   D0  01F9  6895 6$:     MOVL            #14,R12         ; load number of bytes to check
                                     01FC  6896         $DS_BREAK
              00010058 9F   6E   FA  01FC                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16  0203  6897         JSB     CHECK_PACKET            ; go check end packet
                                     0209  6898         $DS_BNCOMPLETE  7$              ; br = no error
                         21 50   E9  0209                       BLBC    R0, 7$
                                     020C  6899         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_INIT2_ERR,PRLINK=DATA_ERR_END
                   00001E9D'EF   DF  020C                       PUSHAL  DATA_ERR_END
                   00001865'EF   DF  0212                       PUSHAL  T_INIT2_ERR
                            00   DD  0218                       PUSHL   #0
                            03   DD  021A                       PUSHL   #$ER
                                     021C       ;;; TEST 11, SUBTEST 1, ERROR 3
              000100D0 9F   04   FB  021C                       CALLS   #$$M, @#DS$ERRHARD
                                     0223  6900         $DS_BCOMPLETE   7$              ; continue if only data error
                         07 50   E8  0223                       BLBS    R0, 7$
                                     0226  6901         $DS_ABORT
              00010020 9F   6C   FA  0226                       CALLG   (AP), @#DS$ABORT
                                     022D  6902 7$:
                                     022D  6903
                                     022D  6904         $DS_ENDSUB
                                     022D       T11_S1_X:
     00010038 9F   000000CC'EF   FA  022D                       CALLG   $S$, @#DS$ENDSUB
                                     0238  6905
                                     0238  6906         $DS_PAGE
```

```
0238                 .SBTTL  SUBTEST 11.2:    SELF DIAGNOSTIC
0238       ;+
0238       ; SUBTEST DESCRIPTION:
0238       ;
0238       ;       This subtest sends the TU58 a command packet to run its internal
0238       ;       diagnostic. While sending each byte of the packet,a timer is set which
0238       ;       will time out if the ready bit in the transmitter processor register
0238       ;       does not set within the alloted time. The same is done for the done
0238       ;       bit in the receiver processor register while reading a byte from the
0238       ;       TU58. When this is completed successfully, the returned end packet is
0238       ;       compared to an expected end packet. If a time out occurs or bad data
0238       ;       is encountered an error will be reported.
0238       ;
0238       ;
0238       ; SUBTEST STEPS:
0238       ;
0238       ;       SUBTEST SELF DIAGNOSTIC
0238       ;       :  Clear buffer
0238       ;       :  Point to self diagnostic command packet
0238       ;       :  REPEAT
0238       ;       :  :  Send command packet byte at a time
0238       ;       :  :  Set timer
0238       ;       :  :  IF timer expired AND no response
0238       ;       :  :  :  THEN
0238       ;       :  :  :  Report no response error
0238       ;       :  :  :  Exit test
0238       ;       :  :  ENDIF
0238       ;       :  : UNTIL all bytes sent
0238       ;       :  ENDREPEAT
0238       ;       :  REPEAT
0238       ;       :  :  Set timer
0238       ;       :  :  IF timer expired AND done set
0238       ;       :  :  :  THEN
0238       ;       :  :  :  :  Read byte
0238       ;       :  :  ENDIF
0238       ;       :  : UNTIL timer expired AND done clear
0238       ;       :  ENDREPEAT
0238       ;       :  IF expected byte count NEQ actual byte count
0238       ;       :  :  THEN
0238       ;       :  :  :  Report error
0238       ;       :  :  :  Exit test
0238       ;       :  ENDIF
0238       ;       :  Point to expected and actual data
0238       ;       :  REPEAT
0238       ;       :  :  IF expected data NEQ received data
0238       ;       :  :  :  THEN
0238       ;       :  :  :  Report data error
0238       ;       :  :  :  Exit test
0238       ;       :  :  ENDIF
0238       ;       :  : UNTIL all data checked
0238       ;       :  ENDREPEAT
0238       ;       ENDSUBTEST SELF DIAGNOSTIC
0238       ;
0238       ;
0238       ;
```

```
                                   0238  6963 ;.page
                                   0238  6964 ;-
                                   0238  6965          $DS_BGNSUB
                                   0238       T11_S2::
    00010030 9F   000000D8'EF  FA  0238                         CALLG   $S$, @#DS$BGNSUB
                                   0243  6966
                                   0243  6967
                  00000000'EF  94  0243  6968          CLRB     B_FLAGS                     ; initialize flags
             58   00000000'FF  9E  0249  6969          MOVAB    @A_BUFFER_BEGIN,R8          ; load buffer starting address
             59   00000400 8F  D0  0250  6970          MOVL     #1024,R9                    ; do both pages
                  00002134'EF  16  0257  6971          JSB      CLEAR_BUFFER                ; go clear buffers
             54   FFFB6C20 8F  D0  025D  6972          MOVL     #DELAY_TIME1,R4             ; load send packet delay time
             55   00000318'EF  9E  0264  6973          MOVAB    MB_DIAGNOSE_CMD,R5          ; point to diagnose command packet
                       57   0E  D0  026B  6974          MOVL     #14,R7                      ; load transfer byte count
                  0000203D'EF  16  026E  6975          JSB      SEND_TU58                   ; send command packet
          24 00000000'EF   00  E1  0274  6976          BBC      #V_CLOCK_FLAG,B_FLAGS,2$ ; br = no timeout error
                                   027C  6977          $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_SLF_DIAG_ERR,PRLINK=NO_READY
                  00001BEB'EF  DF  027C                         PUSHAL  NO_READY
                  00001888'EF  DF  0282                         PUSHAL  T_SLF_DIAG_ERR
                           00  DD  0288                         PUSHL   #0
                           01  DD  028A                         PUSHL   #$ER
                                   028C       ;;; TEST 11, SUBTEST 2, ERROR 1
       000100D0 9F         04  FB  028C                         CALLS   #$$M, @#DS$ERRHARD
                                   0293  6978          $DS_BNCOMPLETE  1$
                        03 50  E9  0293                         BLBC    R0, 1$
                                   0296  6979          $DS_ABORT        TEST
                           50  D4  0296                         CLRL    R0                      ; SEND A WARNING TO THE SUPERVISOR
                           04  0298                         RET                             ; TERMINATE TEST
                                   0299  6980 1$:      $DS_ABORT
       00010020 9F         6C  FA  0299                         CALLG   (AP), @#DS$ABORT
                                   02A0  6981
       54    FFF3CB00 8F   D0  02A0  6982 2$:      MOVL     #DELAY_TIME3,R4             ; load receive packet delay time
       55     00000000'FF  9E  02A7  6983          MOVAB    @A_BUFFER_BEGIN,R5          ; load buffer address to store data
           57    55    0E  C1  02AE  6984          ADDL3    #14,R5,R7                   ; load last expected address of data
              000020C8'EF  16  02B2  6985          JSB      MRECEIVE_TU58               ; go get data from TU58
       5A     00000660'EF  9E  02B8  6986          MOVAB    MB_END_PACKET_1,R10         ; load address of expected end packet
       5B     00000000'FF  9E  02BF  6987          MOVAB    @A_BUFFER_BEGIN,R11         ; load address of actual end packet
                  57   55  D1  02C6  6988          CMPL     R5,R7                       ; check for expected number of bytes
                       31  13  02C9  6989          BEQL     6$                          ; br = correct number of bytes
       00000000'EF     08  88  02CB  6990          BISB2    #M_END_FLAG,B_FLAGS         ; set end error flag
                                   02D2  6591 4$:      $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_SLF_DIAG_ERR,PRLINK=REC_ERR,-
                                   02D2  6992                            P1=#T_REC_END
              00001D2E'8F  DD  02D2                         PUSHL   #T_REC_END
              00001EB4'EF  DF  02D8                         PUSHAL  REC_ERR
              00001888'EF  DF  02DE                         PUSHAL  T_SLF_DIAG_ERR
                       00  DD  02E4                         PUSHL   #0
                       02  DD  02E6                         PUSHL   #$ER
                                   02E8       ;;; TEST 11, SUBTEST 2, ERROR 2
       000100D0 9F     05  FB  02E8                         CALLS   #$$M, @#DS$ERRHARD
                                   02EF  6993          $DS_BNCOMPLETE  5$
                    03 50  E9  02EF                         BLBC    R0, 5$
                                   02F2  6994          $DS_ABORT        TEST
                       50  D4  02F2                         CLRL    R0                      ; SEND A WARNING TO THE SUPERVISOR
                           04  02F4                         RET                             ; TERMINATE TEST
                                   02F5  6995 5$:      $DS_ABORT
          00010020 9F   6C   FA  02F5                  CALLG   (AP), @#DS$ABORT
                                 02FC  6996
                   5C   0E   D0  02FC  6997 6$:    MOVL    #14,R12                 ; load number of bytes to check
                                 02FF  6998        $DS_BREAK
          00010058 9F   6E   FA  02FF                  CALLG   (SP), @#DS$BREAK
               0000213B'EF   16  0306  6999        JSB     CHECK_PACKET            ; go check end packet
                                 030C  7000        $DS_BNCOMPLETE  7$              ; br = no error
                      21 50  E9  030C                  BLBC    R0, 7$
                                 030F  7001        $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_SLF_DIAG_ERR,PRLINK=DATA_ERR_END
               00001E9D'EF   DF  030F                  PUSHAL  DATA_ERR_END
               00001888'EF   DF  0315                  PUSHAL  T_SLF_DIAG_ERR
                        00   DD  031B                  PUSHL   #0
                        03   DD  031D                  PUSHL   #$ER
                                 031F       ;;; TEST 11, SUBTEST 2, ERROR 3
          000100D0 9F   04   FB  031F                  CALLS   #$$M, @#DS$ERRHARD
                                 0326  7002        $DS_BCOMPLETE   7$              ; continue if only data error
                      07 50  E8  0326                  BLBS    R0, 7$
                                 0329  7003        $DS_ABORT
          00010020 9F   6C   FA  0329                  CALLG   (AP), @#DS$ABORT
                                 0330  7004 7$:
                                 0330  7005
                                 0330  7006        $DS_ENDSUB
                                 0330       T11_S2_X:
 00010038 9F   000000D8'EF   FA  0330                  CALLG   $S$, @#DS$ENDSUB
                                 033B  7007
                                 033B  7008        $DS_PAGE
```

```
033B              .SBTTL  SUBTEST 11.3:   NOP
033B        ;+
033B        ; SUBTEST DESCRIPTION:
033B        ;
033B        ;       This subtest sends the TU58 a command packet to perform a no
033B        ;       operation. While sending each byte of the packet, a timer is set which
033B        ;       will time out if the ready bit in the transmitter processor register
033B        ;       does not set within the alloted time. The same is done for the done
033B        ;       bit in the receiver processor register while reading a byte from the
033B        ;       TU58. When this is completed successfully, the returned end packet is
033B        ;       compared to an expected end packet. If a time out occurs while expecting
033B        ;       a byte transfer or bad data is encountered an error will be reported.
033B        ;
033B        ;
033B        ; SUBTEST STEPS:
033B        ;
033B        ;       SUBTEST NOP
033B        ;       :  Clear buffer
033B        ;       :  Point to no operation command packet
033B        ;       :  REPEAT
033B        ;       :  :  Send command packet byte at a time
033B        ;       :  :  Set timer
033B        ;       :  :  IF timer expired AND no response
033B        ;       :  :  :  THEN
033B        ;       :  :  :  Report no response error
033B        ;       :  :  :  Exit test
033B        ;       :  :  ENDIF
033B        ;       :  : UNTIL all bytes sent
033B        ;       :  ENDREPEAT
033B        ;       :  REPEAT
033B        ;       :  :  Set timer
033B        ;       :  :  IF timer expired AND done set
033B        ;       :  :  :  THEN
033B        ;       :  :  :  :  Read byte
033B        ;       :  :  ENDIF
033B        ;       :  : UNTIL timer expired AND done clear
033B        ;       :  ENDREPEAT
033B        ;       :  IF expected byte count NEQ actual byte count
033B        ;       :  :  THEN
033B        ;       :  :  :  Report error
033B        ;       :  :  :  Exit test
033B        ;       :  ENDIF
033B        ;       :  Point to expected and actual end packet
033B        ;       :  REPEAT
033B        ;       :  :  IF expected data NEQ received data
033B        ;       :  :  :  THEN
033B        ;       :  :  :  Report data error
033B        ;       :  :  :  Exit test
033B        ;       :  :  ENDIF
033B        ;       :  : UNTIL all data checked
033B        ;       :  ENDREPEAT
033B        ;       ENDSUBTEST NOP
033B        ;
033B        ;
033B        ;
```

```
                                 033B  7065 ;.page
                                 033B  7066 ;-
                                 033B  7067          $DS_BGNSUB
                                 033B       T11_S3::
    00010030 9F    000000E4'EF  FA  033B                       CALLG    $S$, @#DS$BGNSUB
                                 0346  7068
                                 0346  7069
               00000000'EF  94  0346  7070          CLRB     B_FLAGS                    ; initialize flags
         58    00000000'FF  9E  034C  7071          MOVAB    @A_BUFFER_BEGIN,R8         ; load buffer starting address
         59    00000400 8F  D0  0353  7072          MOVL     #1024,R9                   ; do both pages
               00002134'EF  16  035A  7073          JSB      CLEAR_BUFFER               ; go clear buffers
         54    FFFB6C20 8F  D0  0360  7074          MOVL     #DELAY_TIME1,R4            ; load send packet delay time
         55    0000030A'EF  9E  0367  7075          MOVAB    MB_NOP_CMD,R5              ; point to init command packet
                    57  0E  D0  036E  7076          MOVL     #14,R7                     ; load transfer byte count
               0000203D'EF  16  0371  7077          JSB      SEND_TU58                  ; send command packet
       24 00000000'EF   00  E1  0377  7078          BBC      #V_CLOCK_FLAG,B_FLAGS,2$ ; br = no timeout error
                                 037F  7079          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_NOP_ERR,PRLINK=NO_READY
               00001BEB'EF  DF  037F                         PUSHAL   NO_READY
               000018AD'EF  DF  0385                         PUSHAL   T_NOP_ERR
                        00  DD  038B                         PUSHL    #0
                        01  DD  038D                         PUSHL    #$ER
                                 038F       ;;; TEST 11, SUBTEST 3, ERROR 1
         000100D0 9F    04  FB  038F                         CALLS    #$$M, @#DS$ERRHARD
                                 0396  7080          $DS_BNCOMPLETE  1$
                     03 50  E9  0396                         BLBC     R0, 1$
                                 0399  7081          $DS_ABORT        TEST
                        50  D4  0399                         CLRL     R0                 ; SEND A WARNING TO THE SUPERVISOR
                        04  039B                             RET                         ; TERMINATE TEST
                                 039C  7082 1$:      $DS_ABORT
         00010020 9F    6C  FA  039C                         CALLG    (AP), @#DS$ABORT
                                 03A3  7083
         54    FFFB6C20 8F  D0  03A3  7084 2$:      MOVL     #DELAY_TIME1,R4            ; load receive packet delay time
         55    00000000'FF  9E  03AA  7085          MOVAB    @A_BUFFER_BEGIN,R5         ; load buffer address to store data
            57    55    0E  C1  03B1  7086          ADDL3    #14,R5,R7                  ; load last expected address of data
               000020C8'EF  16  03B5  7087          JSB      MRECEIVE_TU58              ; go get data from TU58
         5A    00000660'EF  9E  03BB  7088          MOVAB    MB_END_PACKET_1,R10        ; load address of expected end packet
         5B    00000000'FF  9E  03C2  7089          MOVAB    @A_BUFFER_BEGIN,R11        ; load address of actual end packet
                    57  55  D1  03C9  7090          CMPL     R5,R7                      ; check for expected number of bytes
                    31  13  03CC  7091              BEQL     6$                         ; br = correct nbumber of bytes
         00000000'EF    08  88  03CE  7092          BISB2    #M_END_FLAG,B_FLAGS        ; set end error flag
                                 03D5  7093 4$:      $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_NOP_ERR,PRLINK=REC_ERR,-
                                 03D5  7094                           P1=#T_REC_END
               00001D2E'8F  DD  03D5                         PUSHL    #T_REC_END
               00001EB4'EF  DF  03DB                         PUSHAL   REC_ERR
               000018AD'EF  DF  03E1                         PUSHAL   T_NOP_ERR
                        00  DD  03E7                         PUSHL    #0
                        02  DD  03E9                         PUSHL    #$ER
                                 03EB       ;;; TEST 11, SUBTEST 3, ERROR 2
         000100D0 9F    05  FB  03EB                         CALLS    #$$M, @#DS$ERRHARD
                                 03F2  7095          $DS_BNCOMPLETE  5$
                     03 50  E9  03F2                         BLBC     R0, 5$
                                 03F5  7096          $DS_ABORT        TEST
                        50  D4  03F5                         CLRL     R0                 ; SEND A WARNING TO THE SUPERVISOR
                        04  03F7                             RET                         ; TERMINATE TEST
                                 03F8  7097 5$:      $DS_ABORT
```

```
           00010020 9F    6C    FA  03F8                      CALLG   (AP), @#DS$ABORT
                                    03FF   7098
                 5C    0E    D0  03FF   7099 6$:       MOVL    #14,R12                      ; load number of bytes to check
                                    0402   7100            $DS_BREAK
           00010058 9F    6E    FA  0402                      CALLG   (SP), @#DS$BREAK
                0000213B'EF    16  0409   7101            JSB     CHECK_PACKET                 ; go check end packet
                                    040F   7102            $DS_BNCOMPLETE  7$                    ; br = no error
                    21 50    E9  040F                      BLBC    R0, 7$
                                    0412   7103            $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_NOP_ERR,PRLINK=DATA_ERR_END
                00001E9D'EF    DF  0412                      PUSHAL  DATA_ERR_END
                000018AD'EF    DF  0418                      PUSHAL  T_NOP_ERR
                       00    DD  041E                      PUSHL   #0
                       03    DD  0420                      PUSHL   #$ER
                                    0422          ;;; TEST 11, SUBTEST 3, ERROR 3
           000100D0 9F    04    FB  0422                      CALLS   #$$M, @#DS$ERRHARD
                                    0429   7104            $DS_BCOMPLETE   7$                    ; continue if only data error
                    07 50    E8  0429                      BLBS    R0, 7$
                                    042C   7105            $DS_ABORT
           00010020 9F    6C    FA  042C                      CALLG   (AP), @#DS$ABORT
                                    0433   7106 7$:
                                    0433   7107
                                    0433   7108            $DS_ENDSUB
                                    0433          T11_S3_X:
   00010038 9F   000000E4'EF    FA  0433                      CALLG   $S$, @#DS$ENDSUB
                                    043E   7109            $DS_CLRVEC_S    VECTOR=#SCB$L_TIMER
                000000C0 8F    DD  043E                      PUSHL   #SCB$L_TIMER
           00010168 9F    01    FB  0444                      CALLS   #1, @#DS$CLRVEC
                                    044B   7110
                                    044B   7111            $DS_ENDTEST
                 50    01    D0  044B                      MOVL    #1, R0                ; NORMAL EXIT
                                    044E          TEST_011_X::
           00010058 9F    6E    FA  044E                      CALLG   (SP), @#DS$BREAK
                             04  0455                      RET                           ; RETURN TO TEST SEQUENCER
                                    0456   7112
                                    0456   7113            $DS_PAGE
```

```
                                     0456                .SBTTL  TEST 12:  TU58 EXERCISER READ/WRITE (WITH MRSP)
                                 00000000                        .PSECT  TEST_012, PAGE, NOWRT
                                     0028
                                     0028  7116 ;++
                                     0028  7117 ; Test description:
                                     0028  7118 ;
                                     0028  7119 ;        This test exercises the TU58 on the VAX-11/750 by performing
                                     0028  7120 ;        position, read (normal and two versions of modified) and write
                                     0028  7121 ;        (normal and two versions of modified) commands. Zero padding of
                                     0028  7122 ;        blocks and records are also checked for write commands. All end and
                                     0028  7123 ;        data packets are checked for errors: Timers are setup to time out if
                                     0028  7124 ;        a byte transaction between the host and TU58 is not completed within
                                     0028  7125 ;        a reasonable amount of time. If any errors are detected, they are
                                     0028  7126 ;        reported with all available information pertaining to the failure.
                                     0028  7127 ;
                                     0028  7128 ;
                                     0028  7129 ; Assumptions:
                                     0028  7130 ;
                                     0028  7131 ;        This test assumes the scratch tape cartridge used is in good
                                     0028  7132 ;        condition. If a marginal tape is used bits may be dropped during the
                                     0028  7133 ;        read modified test, which reads with an increased threshold and will
                                     0028  7134 ;        consequently report false data errors. Any tape may be used, but
                                     0028  7135 ;        existing data will be destroyed. Before the tape is written, the first
                                     0028  7136 ;        directory will be read. If the directory read is not `scrat.ch , the
                                     0028  7137 ;        operator will be prompted to either overwrite the tape or abort this
                                     0028  7138 ;        test. If overwriting the tape is chosen, an attempt to write
                                     0028  7139 ;        `scrat.ch' in the first directory area will be made. If an operational
                                     0028  7140 ;        error occurs it will be reported, but error detection in this subtest
                                     0028  7141 ;        is limited and the primary function is to give the operator an
                                     0028  7142 ;        opportunity to remove a tape containing valuable data from the drive
                                     0028  7143 ;        and inserting a scratch tape before it is destroyed.
                                     0028  7144 ;
                                     0028  7145 ;
                                     0028  7146 ;.page
                                     0028  7147 ;--
                                     0028  7148          $DS_BGNTEST     <DEFAULT,TU58,ALL>
                                     0028       DATA_012:
                            00000000 0028                        .LONG   0                   ; TEST ARGUMENT TABLE TERMINATOR
                                     002C       TEST_012::
                                0000 002C                        .WORD   ^M<>                ; ENTRY MASK
                                     002E  7149
                                     002E  7150
                                     002E  7151          ; Set Up
                                     002E  7152
                                     002E  7153          $DS_SETVEC_S    VECTOR=#SCB$L_TIMER,SRVADR=TIMER_SERVICE
                           00      DD 002E                       PUSHL   #0
                  00002150'EF      DF 0030                       PUSHAL  TIMER_SERVICE
                  000000C0 8F      DD 0036                       PUSHL   #SCB$L_TIMER
             00010160 9F   03      FB 003C                       CALLS   #3, @#DS$SETVEC
                                     0043  7154
             58   00000000'FF      9E 0043  7155 1$:     MOVAB   @A_BUFFER_BEGIN,R8       ; load buffer starting address
00000013'EF  58   0000042E 8F      C1 004A  7156         ADDL3   #1070,R8,BUF_LIM         ; construct upper buffer limit
             59   00000400 8F      D0 0056  7157         MOVL    #1024,R9                 ; do both pages
                  00002134'EF      16 005D  7158         JSB     CLEAR_BUFFER             ; go clear buffers
                                     0063  7159
```

```
          00000000'EF   94  0063  7160          CLRB    B_FLAGS                   ; init flags
          00000001'EF   94  0069  7161          CLRB    B_FLAGS2                  ; init flags
          00001FCB'EF   16  006F  7162 2$:      JSB     MINIT_SEQUENCE            ; initialize drive
      5A  00000023'EF   9E  0075  7163          MOVAB   B_CONTINUE,R10            ; load expected data address
      5B  00000000'FF   9E  007C  7164          MOVAB   @A_BUFFER_BEGIN,R11       ; load actual data address
   2C 00000000'EF   04  E1  0083  7165          BBC     #V_SEND_FLAG,B_FLAGS,5$ ; br = no send error
   DC 00000001'EF   01  E3  008B  7166          BBCS    #V_REPEAT_SEND,B_FLAGS2,2$ ; br = first try and set flag
                            0093  7167          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_INIT1_ERR,PRLINK=NO_READY
          00001BEB'EF   DF  0093                        PUSHAL  NO_READY
          0000183B'EF   DF  0099                        PUSHAL  T_INIT1_ERR
                   00   DD  009F                        PUSHL   #0
                   01   DD  00A1                        PUSHL   #$ER
                            00A3          ;;; TEST 12, SUBTEST 0, ERROR 1
      000100D0 9F   04  FB  00A3                        CALLS   #$$M, @#DS$ERRHARD
                            00AA  7168          $DS_BNCOMPLETE  3$
                03 50   E9  00AA                        BLBC    R0, 3$
                            00AD  7169          $DS_ABORT       TEST
                   50   D4  00AD                        CLRL    R0                ; SEND A WARNING TO THE SUPERVISOR
                   04       00AF                        RET                       ; TERMINATE TEST
                            00B0  7170 3$:      $DS_ABORT
      00010020 9F   6C  FA  00B0                        CALLG   (AP), @#DS$ABORT
                            00B7  7171
   32 00000000'EF   01  E1  00B7  7172 5$:      BBC     #V_RECEIVE_FLAG,B_FLAGS,9$ ; br = no receive error
   A8 00000001'EF   01  E3  00BF  7173          BBCS    #V_REPEAT_SEND,B_FLAGS2,2$ ; br = first try and set flag
                            00C7  7174 7$:      $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_INIT1_ERR,PRLINK=REC_ERR,-
                            00C7  7175                           P1=#T_REC_CONT
          00001D47'8F   DD  00C7                        PUSHL   #T_REC_CONT
          00001EB4'EF   DF  00CD                        PUSHAL  REC_ERR
          0000183B'EF   DF  00D3                        PUSHAL  T_INIT1_ERR
                   00   DD  00D9                        PUSHL   #0
                   02   DD  00DB                        PUSHL   #$ER
                            00DD          ;;; TEST 12, SUBTEST 0, ERROR 2
      000100D0 9F   05  FB  00DD                        CALLS   #$$M, @#DS$ERRHARD
                            00E4  7176          $DS_BNCOMPLETE  8$
                03 50   E9  00E4                        BLBC    R0, 8$
                            00E7  7177          $DS_ABORT       TEST
                   50   D4  00E7                        CLRL    R0                ; SEND A WARNING TO THE SUPERVISOR
                   04       00E9                        RET                       ; TERMINATE TEST
                            00EA  7178 8$:      $DS_ABORT
      00010020 9F   6C  FA  00EA                        CALLG   (AP), @#DS$ABORT
                            00F1  7179
               5C   01  D0  00F1  7180 9$:      MOVL    #1,R12                    ; load number of bytes to check
                            00F4  7181          $DS_BREAK
      00010058 9F   6E  FA  00F4                        CALLG   (SP), @#DS$BREAK
          0000213B'EF   16  00FB  7182          JSB     CHECK_PACKET              ; go check continue packet
                            0101  7183          $DS_BNCOMPLETE  12$
                39 50   E9  0101                        BLBC    R0, 12$
   0A 00000001'EF   01  E0  0104  7184          BBS     #V_REPEAT_SEND,B_FLAGS2,10$ ; br = first try and set flag
      00000001'EF   02  88  010C  7185          BISB2   #M_REPEAT_SEND,B_FLAGS2
                 FF59   31  0113  7186          BRW     2$
               52   50  D0  0116  7187 10$:     MOVL    R0,R2                     ; save code for error report
                            0119  7188          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_INIT1_ERR,PRLINK=DATA_ERR_CONT
          00001E4D'EF   DF  0119                        PUSHAL  DATA_ERR_CONT
          0000183B'EF   DF  011F                        PUSHAL  T_INIT1_ERR
                   00   DD  0125                        PUSHL   #0
```

```
                            03   DD  0127                       PUSHL   #$ER
                                     0129       ;;; TEST 12, SUBTEST 0, ERROR 3
              000100D0 9F   04   FB  0129                       CALLS   #$$M, @#DS$ERRHARD
                                     0130  7189         $DS_BNCOMPLETE  11$
                         03 50   E9  0130                       BLBC    R0, 11$
                                     0133  7190         $DS_ABORT       TEST
                            50   D4  0133                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  0135                       RET                     ; TERMINATE TEST
                                     0136  7191 11$:    $DS_ABORT
              00010020 9F   6C   FA  0136                       CALLG   (AP), @#DS$ABORT
                                     013D  7192 12$:
                                     013D  7193         $DS_PAGE
```

```
013D              .SBTTL  SUBTEST 12.1:   READ DIRECTORY
013D  ;+
013D  ; SUBTEST DESCRIPTION:
013D  ;
013D  ;       This subtest sends the TU58 a command packet to read the first entry
013D  ;       in the directory area of the tape. A timer is set which will
013D  ;       time out if the ready bit in the transmitter processor register does
013D  ;       not set within the allotted time. A second timer is set which will
013D  ;       time out if the done bit in the receiver processor register does not
013D  ;       set within the allotted amount of time. If either of these timers
013D  ;       expires while expecting a data transfer an error will be reported
013D  ;       stating the conditions existing at the time of failure.
013D  ;       The returned data packet is examined to see if the directory on the
013D  ;       tape is 'scrat.ch'. If it is a scratch tape testing continues; if not,
013D  ;       the operator will be prompted to either abort the TU58 exerciser test
013D  ;       or overwrite the tape in the drive. Before entering the appropriate
013D  ;       response on the TTY, the tape cartridge may be replaced with a scratch
013D  ;       tape. In the case of an overwrite, the TU58 will be sent a write
013D  ;       command  packet to write 'scrat.ch' in the directory.  Timers are set
013D  ;       which will time out if the appropriate bits do not set within allotted
013D  ;       times. If an error occurs all appropriate information will be
013D  ;       reported. This subtest is not primarily used for error detection, if an
013D  ;       error is encountered, testing should continue with the next subtest
013D  ;       for diagnosis.
013D  ;
013D  ;
013D  ; SUBTEST STEPS:
013D  ;
013D  ;       SUBTEST READ DIRECTORY
013D  ;       :  Clear buffer
013D  ;       :  Point to read command packet
013D  ;       :  REPEAT
013D  ;       :  :  Send byte of command packet
013D  ;       :  :  Set timer
013D  ;       :  :  IF no response AND timer expired
013D  ;       :  :  :  THEN
013D  ;       :  :  :  Report no response error
013D  ;       :  :  :  Exit test
013D  ;       :  :  ENDIF
013D  ;       :  : UNTIL all bytes sent
013D  ;       :  ENDREPEAT
013D  ;       :  REPEAT
013D  ;       :  :  Set timer
013D  ;       :  :  IF timer expired AND done set
013D  ;       :  :  :  THEN
013D  ;       :  :  :  :  Read byte
013D  ;       :  :  ENDIF
013D  ;       :  : UNTIL timer expired AND done clear
013D  ;       :  ENDREPEAT
013D  ;       :  IF expected byte count NEQ actual byte count
013D  ;       :  :  THEN
013D  ;       :  :  :  Report error
013D  ;       :  :  :  Exit test
013D  ;       :  ENDIF
013D  ;       :  Examine data for first directory on tape
```

```
013D  ;     : IF directory NEQ scrat.ch
013D  ;     : : THEN
013D  ;     : : WHILE write directory flag EQL zero DO
013D  ;     : : : Prompt operator to overwrite directory
013D  ;     : : : IF response EQL no
013D  ;     : : : : THEN
013D  ;     : : : : Exit test
013D  ;     : : : ENDIF
013D  ;     : : : IF response EQL yes
013D  ;     : : : : THEN
013D  ;     : : : : Set write directory flag
013D  ;     : : : : Point to write directory command packet
013D  ;     : : : : REPEAT
013D  ;     : : : : : Send command packet byte at a time
013D  ;     : : : : : Set timer
013D  ;     : : : : : IF timer expired AND no response
013D  ;     : : : : : : THEN
013D  ;     : : : : : : Report no response error
013D  ;     : : : : : : Exit test
013D  ;     : : : : : ENDIF
013D  ;     : : : : : UNTIL all bytes sent
013D  ;     : : : : ENDREPEAT
013D  ;     : : : : Wait for continue
013D  ;     : : : : Set timer
013D  ;     : : : : IF timer expired AND no response
013D  ;     : : : : : THEN
013D  ;     : : : : : Report no response error
013D  ;     : : : : : Exit test
013D  ;     : : : : ENDIF
013D  ;     : : : : Point to data packet with  scrat.ch' directory
013D  ;     : : : : REPEAT
013D  ;     : : : : : Send data packet byte at a time
013D  ;     : : : : : Set timer
013D  ;     : : : : : IF timer expired AND no response
013D  ;     : : : : : : THEN
013D  ;     : : : : : : Report no response error
013D  ;     : : : : : : Exit test
013D  ;     : : : : : ENDIF
013D  ;     : : : : : UNTIL all bytes sent
013D  ;     : : : : ENDREPEAT
013D  ;     : : : : REPEAT
013D  ;     : : : : : Read end packet into buffer byte at a t'me
013D  ;     : : : : : Set timer
013D  ;     : : : : : IF timer expired AND no response
013D  ;     : : : : : : THEN
013D  ;     : : : : : : Report no response error
013D  ;     : : : : : : Exit test
013D  ;     : : : : : ENDIF
013D  ;     : : : : : : UNTIL all bytes read
013D  ;     : : : : ENDREPEAT
013D  ;     : : : : Point to expected and received end packets
013D  ;     : : : : REPEAT
013D  ;     : : : : : IF expected data NEQ received data
013D  ;     : : : : : : THEN
013D  ;     : : : : : : Report data error
```

```
                                013D  7305 ;       :  :  :  :  :  :  Exit subtest
                                013D  7306 ;       :  :  :  :  :  ENDIF
                                013D  7307 ;       :  :  :  :  : UNTIL all bytes checked
                                013D  7308 ;       :  :  :  :  ENDREPEAT
                                013D  7309 ;       :  :  :  ENDIF
                                013D  7310 ;       :  :  ENDWHILE
                                013D  7311 ;       :  ENDIF
                                013D  7312 ;       ENDSUBTEST READ DIRECTORY
                                013D  7313 ;
                                013D  7314 ;
                                013D  7315 ;
                                013D  7316 ;.page
                                013D  7317 ;-
                                013D  7318
                                013D  7319
                                013D  7320         $DS_BGNSUB
                                013D       T12_S1::
00010030 9F     000000F0'EF   FA  013D                     CALLG   $S$, @#DS$BGNSUB
                                0148  7321                                          ; clear buffers
              00000000'EF   94  0148  7322         CLRB    B_FLAGS                  ; initialize flags
         58   00000000'FF   9E  014E  7323         MOVAB   @A_BUFFER_BEGIN,R8       ; load buffer starting address
         59   00000400 8F   D0  0155  7324         MOVL    #1024,R9                 ; do two pages
              00002134'EF   16  015C  7325         JSB     CLEAR_BUFFER             ; go clear buffers
                                0162  7326                                          ; send read command packet
         54   FF7E4CE0 8F   D0  0162  7327         MOVL    #DELAY_TIME7,R4          ; load send packet delay time
         55   00000326'EF   9E  0169  7328         MOVAB   MB_READ_1_CMD,R5                 ; point to read command packet
                  57   0E   D0  0170  7329         MOVL    #14,R7                   ; load transfer byte count
              0000203D'EF   16  0173  7330         JSB     SEND_TU58                ; send command packet
      24 00000000'EF   00   E1  0179  7331         BBC     #V_CLOCK_FLAG,B_FLAGS,2$ ; br = no timeout error
                                0181  7332         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=NO_READY
              00001BEB'EF   DF  0181                       PUSHAL  NO_READY
              00001931'EF   DF  0187                       PUSHAL  T_RD_REC_ERR
                       00   DD  018D                       PUSHL   #0
                       01   DD  018F                       PUSHL   #$ER
                                0191       ;;; TEST 12, SUBTEST 1, ERROR 1
         000100D0 9F   04   FB  0191                       CALLS   #$$M, @#DS$ERRHARD
                                0198  7333         $DS_BNCOMPLETE  1$
                    03 50   E9  0198                       BLBC    R0, 1$
                                019B  7334         $DS_ABORT       TEST
                       50   D4  019B                       CLRL    R0               ; SEND A WARNING TO THE SUPERVISOR
                       04   04  019D                       RET                      ; TERMINATE TEST
                                019E  7335 1$:     $DS_ABORT
         00010020 9F   6C   FA  019E                       CALLG   (AP), @#DS$ABORT
                                01A5  7336                                          ; read data and end packet
         54   FAEF00C0 8F   D0  01A5  7337 2$:     MOVL    #DELAY_TIME8,R4          ; load receive packet delay time
         55   00000000'FF   9E  01AC  7338         MOVAB   @A_BUFFER_BEGIN,R5       ; load buffer address to store data
    57   55   00000092 8F   C1  01B3  7339         ADDL3   #146,R5,R7               ; load last expected address of data
              000020C8'EF   16  01BB  7340         JSB     MRECEIVE_TU58            ; go get data from TU58
                  57   55   D1  01C1  7341         CMPL    R5,R7                    ; check for correct number of bytes
                       24   13  01C4  7342         BEQL    6$                       ; br = correct number of bytes
                                01C6  7343         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=MSG_1
              00001AD6'EF   DF  01C6                       PUSHAL  MSG_1
              00001931'EF   DF  01CC                       PUSHAL  T_RD_REC_ERR
                       00   DD  01D2                       PUSHL   #0
                       02   DD  01D4                       PUSHL   #$ER
```

```
                                     01D6       ;;; TEST 12, SUBTEST 1, ERROR 2
              000100D0 9F   04   FB  01D6                       CALLS   #$$M, @#DS$ERRHARD
                                     01DD  7344         $DS_BNCOMPLETE  4$
                         03 50   E9  01DD                       BLBC    R0, 4$
                                     01E0  7345         $DS_ABORT         TEST
                            50   D4  01E0                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  01E2                       RET                     ; TERMINATE TEST
                                     01E3  7346
                                     01E3  7347 4$:     $DS_ABORT
              00010020 9F   6C   FA  01E3                       CALLG   (AP), @#DS$ABORT
                                     01EA  7348                                         ; check end packet
              5A   0000066E'EF   9E  01EA  7349 6$:     MOVAB   MB_END_PACKET_2,R10     ; load address of expected end packet
              5B   00000000'FF   9E  01F1  7350         MOVAB   @A_BUFFER_BEGIN,R11     ; load address of actual end packet
              5B   00000084 8F   C0  01F8  7351         ADDL2   #132,R11                ; skip over data packet in buffer
                       5C   0E   D0  01FF  7352         MOVL    #14,R12                 ; load number of bytes to check
                                     0202  7353         $DS_BREAK
              00010058 9F   6E   FA  0202                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16  0209  7354         JSB     CHECK_PACKET            ; go check end packet
                                     020F  7355         $DS_BNCOMPLETE  7$              ; br = no error
                         21 50   E9  020F                       BLBC    R0, 7$
                                     0212  7356         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=DATA_ERR_END
                   00001E9D'EF   DF  0212                       PUSHAL  DATA_ERR_END
                   00001931'EF   DF  0218                       PUSHAL  T_RD_REC_ERR
                            00   DD  021E                       PUSHL   #0
                            03   DD  0220                       PUSHL   #$ER
                                     0222       ;;; TEST 12, SUBTEST 1, ERROR 3
              000100D0 9F   04   FB  0222                       CALLS   #$$M, @#DS$ERRHARD
                                     0229  7357         $DS_BCOMPLETE   7$              ; br = data error, but continue
                         07 50   E8  0229                       BLBS    R0, 7$
                                     022C  7358         $DS_ABORT
              00010020 9F   6C   FA  022C                       CALLG   (AP), @#DS$ABORT
                                     0233  7359                                         ; check first directory entry
              5A   000006EC'EF   9E  0233  7360 7$:     MOVAB   B_DATA_PKT_1,R10        ; load address of expected data packet
                       5A   0E   C0  023A  7361         ADDL2   #14,R10                 ; point to expected first directory entry
              5B   00000000'FF   9E  023D  7362         MOVAB   @A_BUFFER_BEGIN,R11     ; load address of actual data packet
                       5B   0E   C0  0244  7363         ADDL2   #14,R11                 ; point to actual first directory entry
                       5C   06   D0  0247  7364         MOVL    #6,R12                  ; do only directory entry
                                     024A  7365         $DS_BREAK
              00010058 9F   6E   FA  024A                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16  0251  7366         JSB     CHECK_PACKET            ; go check data packet
                                     0257  7367         $DS_BCOMPLETE   8$              ; br =  unknown tape
                         03 50   E8  0257                       BLBS    R0, 8$
                                     025A  7368         $DS_EXIT        SUB             ; exit subtest if scratch tape
                          01E5   31  025A                       BRW     T12_S1_X        ; EXIT SUBTEST 1
                                     025D  7369 8$:     $DS_ASKLGCL_S   T_OVERWRITE,B_RESPONSE,,9$,0,
                            00   DD  025D                       PUSHL   #0
                            00   DD  025F                       PUSHL   #PAR$_NO
                            00   DD  0261                       PUSHL   #0
                   00000284'EF   DF  0263                       PUSHAL  9$
                            00   DD  0269                       PUSHL   #0
                   00000012'EF   DF  026B                       PUSHAL  B_RESPONSE
                   00001B56'EF   DF  0271                       PUSHAL  T_OVERWRITE
              00010098 9F   07   FB  0277                       CALLS   #7, @#DS$ASKLGCL
                                     027E  7370         $DS_EXIT        TEST            ; exit test if negative response
                       50   01   D0  027E                       MOVL    #1,R0           ; NOT AN ERROR
```

```
                  1EC6    31  0281                          BRW     TEST_012_X          ; EXIT TEST 12
                              0284   7371                                               ; positive response, write directory
        54   FF7E4CE0 8F  D0  0284   7372 9$:        MOVL    #DELAY_TIME7,R4            ; load send packet delay time
        55   00000334'EF  9E  028B   7373            MOVAB   MB_WRITE_1_CMD,R5          ; point to write command packet
                  57  0E  D0  0292   7374            MOVL    #14,R7                     ; load transfer byte count
             0000203D'EF  16  0295   7375            JSB     SEND_TU58                  ; send command packet
     24 00000000'EF   00  E1  029B   7376            BBC     #V_CLOCK_FLAG,B_FLAGS,20$ ; br = no timeout error
                              02A3   7377            $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=NO_READY
             00001BEB'EF  DF  02A3                          PUSHAL  NO_READY
             0000190D'EF  DF  02A9                          PUSHAL  T_WR_REC_ERR
                      00  DD  02AF                          PUSHL   #0
                      04  DD  02B1                          PUSHL   #SER
                              02B3          ;;; TEST 12, SUBTEST 1, ERROR 4
        000100D0 9F   04  FB  02B3                          CALLS   #$$M, @#DS$ERRHARD
                              02BA   7378            $DS_BNCOMPLETE  10$
                   03 50  E9  02BA                          BLBC    R0, 10$
                              02BD   7379            $DS_ABORT       TEST
                      50  D4  02BD                          CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                      04  02BF                              RET                         ; TERMINATE TEST
                              02C0   7380 10$:       $DS_ABORT
        00010020 9F   6C  FA  02C0                          CALLG   (AP), @#DS$ABORT
                              02C7   7381                                               ; read continue
        54   FFF3CB00 8F  D0  02C7   7382 20$:       MOVL    #DELAY_TIME3,R4            ; load receive packet delay time
        55   00000000'FF  9E  02CE   7383            MOVAB   @A_BUFFER_BEGIN,R5         ; load buffer address to store data
        55   00000096 8F  C0  02D5   7384            ADDL2   #150,R5                    ; get above previous data
           57   55    01  C1  02DC   7385            ADDL3   #1,R5,R7                   ; load last expected address of data
             000020C8'EF  16  02E0   7386            JSB     MRECEIVE_TU58              ; go get data from TU58
        5A   00000023'EF  9E  02E6   7387            MOVAB   B_CONTINUE,R10             ; load address of expected data
        5B   00000000'FF  9E  02ED   7388            MOVAB   @A_BUFFER_BEGIN,R11        ; load address of buffer
        5B   00000096 8F  C0  02F4   7389            ADDL    #150,R11                   ; point to address of actual data
                  57  55  D1  02FB   7390            CMPL    R5,R7                      ; check for correct number of bytes
                      2A  13  02FE   7391            BEQL    60$                        ; br = correct number of bytes
                              0300   7392 25$:       $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=REC_ERR,-
                              0300   7393                             P1=#T_REC_CONT
             00001D47'8F  DD  0300                          PUSHL   #T_REC_CONT
             00001EB4'EF  DF  0306                          PUSHAL  REC_ERR
             0000190D'EF  DF  030C                          PUSHAL  T_WR_REC_ERR
                      00  DD  0312                          PUSHL   #0
                      05  DD  0314                          PUSHL   #SER
                              0316          ;;; TEST 12, SUBTEST 1, ERROR 5
        000100D0 9F   05  FB  0316                          CALLS   #$$M, @#DS$ERRHARD
                              031D   7394            $DS_BNCOMPLETE  30$
                   03 50  E9  031D                          BLBC    R0, 30$
                              0320   7395            $DS_ABORT       TEST
                      50  D4  0320                          CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                      04  0322                              RET                         ; TERMINATE TEST
                              0323   7396
                              0323   7397 30$:       $DS_ABORT
        00010020 9F   6C  FA  0323                          CALLG   (AP), @#DS$ABORT
                              032A   7398                                               ; check continue packet
                  5C  01  D0  032A   7399 60$:       MOVL    #1,R12                     ; load number of bytes to check
                              032D   7400            $DS_BREAK
        00010058 9F   6E  FA  032D                          CALLG   (SP), @#DS$BREAK
             0000213B'EF  16  0334   7401            JSB     CHECK_PACKET               ; go check continue
                              033A   7402            $DS_BNCOMPLETE  75$                ; br = no error
             24 50    E9  033A                   BLBC    R0, 75$
                          033D  7403        $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=DATA_ERR_DATA
      00001E75'EF     DF  033D                   PUSHAL  DATA_ERR_DATA
      0000190D'EF     DF  0343                   PUSHAL  T_WR_REC_ERR
               00     DD  0349                   PUSHL   #0
               06     DD  034B                   PUSHL   #$ER
                          034D          ;;; TEST 12, SUBTEST 1, ERROR 6
  000100D0 9F  04     FB  034D                   CALLS   #$$M, @#DS$ERRHARD
                          0354  7404        $DS_BNCOMPLETE  70$                  ; br = error, but continue
             03 50    E9  0354                   BLBC    R0, 70$
                          0357  7405        $DS_ABORT        TEST
               50     D4  0357                   CLRL    R0                   ; SEND A WARNING TO THE SUPERVISOR
               04     04  0359                   RET                          ; TERMINATE TEST
                          035A  7406 70$:   $DS_ABORT
  00010020 9F  6C     FA  035A                   CALLG   (AP), @#DS$ABORT
                          0361  7407                                          ; send data packet
  54    FFFB6C20 8F   D0  0361  7408 75$:   MOVL    #DELAY_TIME1,R4              ; load send packet delay time
  55    000006EC'EF   9E  0368  7409        MOVAB   B_DATA_PKT_1,R5              ; point to data packet
           57   1E    D0  036F  7410        MOVL    #30,R7                       ; load transfer byte count
      0000203D'EF     16  0372  7411        JSB     SEND_TU58                    ; send data packet
24 00000000'EF   00   E1  0378  7412        BBC     #V_CLOCK_FLAG,B_FLAGS,90$ ; br = no timeout error
                          0380  7413        $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=NO_READY
      00001BEB'EF     DF  0380                   PUSHAL  NO_READY
      0000190D'EF     DF  0386                   PUSHAL  T_WR_REC_ERR
               00     DD  038C                   PUSHL   #0
               07     DD  038E                   PUSHL   #$ER
                          0390          ;;; TEST 12, SUBTEST 1, ERROR 7
  000100D0 9F  04     FB  0390                   CALLS   #$$M, @#DS$ERRHARD
                          0397  7414        $DS_BNCOMPLETE  80$
             03 50    E9  0397                   BLBC    R0, 80$
                          039A  7415        $DS_ABORT        TEST
               50     D4  039A                   CLRL    R0                   ; SEND A WARNING TO THE SUPERVISOR
               04     04  039C                   RET                          ; TERMINATE TEST
                          039D  7416 80$:   $DS_ABORT
  00010020 9F  6C     FA  039D                   CALLG   (AP), @#DS$ABORT
                          03A4  7417                                          ; read end packet
  54    FF7E4CE0 8F   D0  03A4  7418 90$:   MOVL    #DELAY_TIME7,R4              ; load read packet delay time
  55    00000000'FF   9E  03AB  7419        MOVAB   @A_BUFFER_BEGIN,R5           ; load buffer address to store data
  55    000000A8 8F   C0  03B2  7420        ADDL2   #168,R5                      ; get above existing data
     57   55    0E    C1  03B9  7421        ADDL3   #14,R5,R7                    ; load last expected address of data
      000020C8'EF     16  03BD  7422        JSB     MRECEIVE_TU58                ; go get data from TU58
  5A    0000067C'EF   9E  03C3  7423        MOVAB   MB_END_PACKET_3,R10          ; load address of expected end packet
  5B    00000000'FF   9E  03CA  7424        MOVAB   @A_BUFFER_BEGIN,R11          ; load address of buffer
  5B    000000A8 8F   C0  03D1  7425        ADDL2   #168,R11                     ; load address of actual end packet
           57   55    D1  03D8  7426        CMPL    R5,R7                        ; check for correct number of bytes
               31     13  03DB  7427        BEQL    130$                         ; br = correct number of bytes
  00000000'EF  08     88  03DD  7428        BISB2   #M_END_FLAG,B_FLAGS          ; set end error flag
                          03E4  7429 95$:   $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=REC_ERR,-
                          03E4  7430                         P1=#T_REC_END
      00001D2E'8F     DD  03E4                   PUSHL   #T_REC_END
      00001EB4'EF     DF  03EA                   PUSHAL  REC_ERR
      0000190D'EF     DF  03F0                   PUSHAL  T_WR_REC_ERR
               00     DD  03F6                   PUSHL   #0
               08     DD  03F8                   PUSHL   #$ER
                          03FA          ;;; TEST 12, SUBTEST 1, ERROR 8
           000100D0 9F    05   FB  03FA                         CALLS    #$$M, a#DS$ERRHARD
                                   0401  7431         $DS_BNCOMPLETE  100$
                  03 50   E9  0401                         BLBC     R0, 100$
                                   0404  7432         $DS_ABORT       TEST
                        50   D4  0404                         CLRL     R0                  ; SEND A WARNING TO THE SUPERVISOR
                             04  0406                         RET                          ; TERMINATE TEST
                                   0407  7433
                                   0407  7434 100$:   $DS_ABORT
           00010020 9F    6C   FA  0407                         CALLG    (AP), a#DS$ABORT
                                   040E  7435                                              ; check end packet
                    5C    0E   D0  040E  7436 130$:   MOVL    #14,R12                      ; load number of bytes to check
                                   0411  7437         $DS_BREAK
           00010058 9F    6E   FA  0411                         CALLG    (SP), a#DS$BREAK
               0000213B'EF    16  0418  7438         JSB     CHECK_PACKET                  ; go check end packet
                                   041E  7439         $DS_BNCOMPLETE  140$                 ; br = no error
                  21 50   E9  041E                         BLBC     R0, 140$
                                   0421  7440         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=DATA_ERR_END
               00001E9D'EF    DF  0421                         PUSHAL   DATA_ERR_END
               0000190D'EF    DF  0427                         PUSHAL   T_WR_REC_ERR
                        00   DD  042D                         PUSHL    #0
                        09   DD  042F                         PUSHL    #$ER
                                   0431         ;;; TEST 12, SUBTEST 1, ERROR 9
           000100D0 9F    04   FB  0431                         CALLS    #$$M, a#DS$ERRHARD
                                   0438  7441         $DS_BCOMPLETE   140$
                  07 50   E8  0438                         BLBS     R0, 140$
                                   043B  7442         $DS_ABORT
           00010020 9F    6C   FA  043B                         CALLG    (AP), a#DS$ABORT
                                   0442  7443 140$:
                                   0442  7444
                                   0442  7445
                                   0442  7446
                                   0442  7447         $DS_ENDSUB
                                   0442         T12_S1_X:
  00010038 9F    000000F0'EF    FA  0442                         CALLG    $S$, a#DS$ENDSUB
                                   044D  7448
                                   044D  7449         $DS_PAGE
```

```
044D                  .SBTTL  SUBTEST 12.2:   WRITE BLOCK
044D        ;+
044D        ; SUBTEST DESCRIPTION:
044D        ;
044D        ;       This subtest sends the TU58 a command packet to write one block
044D        ;       with 512 bytes of data. A continue flag should be returned between
044D        ;       sending each data packet and an end packet is then returned from the
044D        ;       TU58. Four 128-byte data packets are sent to fill one block. All byte
044D        ;       transactions between the host and TU58 are timed. If any timer
044D        ;       times out while expecting a data transfer to occur an error
044D        ;       is reported. The data packets contain alternating bytes of all
044D        ;       ones and all zeroes. The end packet is checked and if any errors
044D        ;       have occured, they are reported.
044D        ;
044D        ;
044D        ; SUBTEST STEPS:
044D        ;
044D        ;       SUBTEST WRITE BLOCK
044D        ;       :  Clear buffer
044D        ;       :  Point to write command packet
044D        ;       :  REPEAT
044D        ;       :  :  Send command packet byte at a time
044D        ;       :  :  Set timer
044D        ;       :  :  IF timer expired AND no response
044D        ;       :  :  :  THEN
044D        ;       :  :  :  Report no response error
044D        ;       :  :  :  Exit test
044D        ;       :  :  ENDIF
044D        ;       :  : UNTIL all bytes sent
044D        ;       :  ENDREPEAT
044D        ;       :  DO for count = 1 to 4
044D        ;       :  :  Wait for continue
044D        ;       :  :  Set timer
044D        ;       :  :  IF timer expired AND no response
044D        ;       :  :  :  THEN
044D        ;       :  :  :  Report no response error
044D        ;       :  :  :  Exit test
044D        ;       :  :  ENDIF
044D        ;       :  :  Point to data packet
044D        ;       :  :  REPEAT
044D        ;       :  :  :  Send data packet byte at a time
044D        ;       :  :  :  Set timer
044D        ;       :  :  :  IF timer expired AND no response
044D        ;       :  :  :  :  THEN
044D        ;       :  :  :  :  Report no response error
044D        ;       :  :  :  :  Exit test
044D        ;       :  :  :  ENDIF
044D        ;       :  :  : UNTIL all bytes sent
044D        ;       :  :  ENDREPEAT
044D        ;       :  ENDDO
044D        ;       :  REPEAT
044D        ;       :  :  Set timer
044D        ;       :  :  IF timer expired AND done set
044D        ;       :  :  :  THEN
044D        ;       :  :  :  :  Read byte
```

```
                                     044D  7506 ;       :  :  ENDIF
                                     044D  7507 ;       :  : UNTIL timer expired AND done clear
                                     044D  7508 ;       :  ENDREPEAT
                                     044D  7509 ;       :  IF expected byte count NEQ actual byte count
                                     044D  7510 ;       :  :  THEN
                                     044D  7511 ;       :  :  :  Report error
                                     044D  7512 ;       :  :  :  Exit test
                                     044D  7513 ;       :  ENDIF
                                     044D  7514 ;       :  REPEAT
                                     044D  7515 ;       :  :  Point to expected end packet
                                     044D  7516 ;       :  :  IF expected end packet data NEQ received end packet data
                                     044D  7517 ;       :  :  :  THEN
                                     044D  7518 ;       :  :  :  Report end packet data error
                                     044D  7519 ;       :  :  ENDIF
                                     044D  7520 ;       :  : UNTIL all data checked
                                     044D  7521 ;       :  ENDREPEAT
                                     044D  7522 ;       ENDSUBTEST WRITE BLOCK
                                     044D  7523 ;
                                     044D  7524 ;
                                     044D  7525 ;
                                     044D  7526 ;.page
                                     044D  7527 ;-
                                     044D  7528         $DS_BGNSUB
                                     044D       T12_S2::
     00010030 9F     000000FC'EF   FA  044D                       CALLG   $S$, @#DS$BGNSUB
                                     0458  7529                                         ; clear buffers
                   00000000'EF   94  0458  7530         CLRB    B_FLAGS                 ; initialize flags
              58   00000000'FF   9E  045E  7531         MOVAB   @A_BUFFER_BEGIN,R8      ; load buffer starting address
              59   00000400 8F   D0  0465  7532         MOVL    #1024,R9                ; do two pages
                   00002134'EF   16  046C  7533         JSB     CLEAR_BUFFER            ; go clear buffers
                                     0472  7534                                         ; send write command
              54   FFF3CB00 8F   D0  0472  7535         MOVL    #DELAY_TIME3,R4         ; load send packet delay time
              55   00000342'EF   9E  0479  7536         MOVAB   MB_WRITE_2_CMD,R5       ; point to write command packet
                      57    0E   D0  0480  7537         MOVL    #14,R7                  ; load transfer byte count
                   0000203D'EF   16  0483  7538         JSB     SEND_TU58               ; send command packet
           24 00000000'EF   00   E1  0489  7539         BBC     #V_CLOCK_FLAG,B_FLAGS,20$ ; br = no timeout error
                                     0491  7540         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=NO_READY
                   00001BEB'EF   DF  0491                       PUSHAL  NO_READY
                   000018C8'EF   DF  0497                       PUSHAL  T_WR_BLK_ERR
                            00   DD  049D                       PUSHL   #0
                            01   DD  049F                       PUSHL   #$ER
                                     04A1       ;;; TEST 12, SUBTEST 2, ERROR 1
              000100D0 9F   04   FB  04A1                       CALLS   #$$M, @#DS$ERRHARD
                                     04A8  7541         $DS_BNCOMPLETE  10$
                         03 50   E9  04A8                       BLBC    R0, 10$
                                     04AB  7542         $DS_ABORT       TEST
                            50   D4  04AB                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                            04       04AD                       RET                     ; TERMINATE TEST
                                     04AE  7543 10$:    $DS_ABORT
              00010020 9F   6C   FA  04AE                       CALLG   (AP), @#DS$ABORT
                                     04B5  7544                                         ; read continue
                   00000003'EF   94  04B5  7545 20$:    CLRB    B_COUNT                 ; clear counter to send 4 data packets
              54   FFC2F700 8F   D0  04BB  7546 30$:    MOVL    #DELAY_TIME6,R4         ; load receive packet delay time
              55   00000000'FF   9E  04C2  7547         MOVAB   @A_BUFFER_BEGIN,R5      ; load buffer address to store data
                  57    55   01   C1  04C9  7548         ADDL3   #1,R5,R7                ; load last expected address of data
```

```
          000020C8'EF   16  04CD  7549          JSB     MRECEIVE_TU58              ; go get data from TU58
     5A   00000023'EF   9E  04D3  7550          MOVAB   B_CONTINUE,R10             ; load address of expected data
     5B   00000000'FF   9E  04DA  7551          MOVAB   @A_BUFFER_BEGIN,R11        ; load address of buffer
                57  55  D1  04E1  7552          CMPL    R5,R7                      ; check for correct number of bytes
                    2A  13  04E4  7553          BEQL    70$                        ; br = correct number of bytes
                            04E6  7554 35$:     $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=REC_ERR,-
                            04E6  7555                           P1=#T_REC_CONT
          00001D47'8F   DD  04E6                         PUSHL   #T_REC_CONT
          00001EB4'EF   DF  04EC                         PUSHAL  REC_ERR
          000018C8'EF   DF  04F2                         PUSHAL  T_WR_BLK_ERR
                    00  DD  04F8                         PUSHL   #0
                    02  DD  04FA                         PUSHL   #$ER
                            04FC        ;;; TEST 12, SUBTEST 2, ERROR 2
      000100D0 9F   05  FB  04FC                         CALLS   #$$M, @#DS$ERRHARD
                            0503  7556          $DS_BNCOMPLETE   40$
                 03 50  E9  0503                         BLBC    R0, 40$
                            0506  7557          $DS_ABORT        TEST
                    50  D4  0506                         CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                    04  0508                             RET                         ; TERMINATE TEST
                            0509  7558
                            0509  7559 40$:     $DS_ABORT
      00010020 9F   6C  FA  0509                         CALLG   (AP), @#DS$ABORT
                            0510  7560                                               ; check continue
             5C     01  D0  0510  7561 70$:     MOVL    #1,R12                     ; load number of bytes to check
                            0513  7562          $DS_BREAK
      00010058 9F   6E  FA  0513                         CALLG   (SP), @#DS$BREAK
         0000213B'EF    16  051A  7563          JSB     CHECK_PACKET               ; go check continue
                            0520  7564          $DS_BNCOMPLETE   90$               ; br = no error
                 24 50  E9  0520                         BLBC    R0, 90$
                            0523  7565          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=DATA_ERR_CONT
          00001E4D'EF   DF  0523                         PUSHAL  DATA_ERR_CONT
          000018C8'EF   DF  0529                         PUSHAL  T_WR_BLK_ERR
                    00  DD  052F                         PUSHL   #0
                    03  DD  0531                         PUSHL   #$ER
                            0533        ;;; TEST 12, SUBTEST 2, ERROR 3
      000100D0 9F   04  FB  0533                         CALLS   #$$M, @#DS$ERRHARD
                            053A  7566          $DS_BNCOMPLETE   80$
                 03 50  E9  053A                         BLBC    R0, 80$
                            053D  7567          $DS_ABORT        TEST
                    50  D4  053D                         CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                    04  053F                             RET                         ; TERMINATE TEST
                            0540  7568 80$:     $DS_ABORT
      00010020 9F   6C  FA  0540                         CALLG   (AP), @#DS$ABORT
                            0547  7569                                               ; send data packet
     54   FFFB6C20 8F   D0  0547  7570 90$:     MOVL    #DELAY_TIME1,R4            ; load send packet delay time
     55   0000070A'EF   9E  054E  7571          MOVAB   B_DATA_PKT_2,R5            ; point to data packet
     57   00000084 8F   D0  0555  7572          MOVL    #132,R7                    ; load transfer byte count
          0000203D'EF   16  055C  7573          JSB     SEND_TU58                  ; send data packet
  24 00000000'EF    00  E1  0562  7574          BBC     #V_CLOCK_FLAG,B_FLAGS,110$ ; br = no timeout error
                            056A  7575          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=NO_READY
          00001BEB'EF   DF  056A                         PUSHAL  NO_READY
          000018C8'EF   DF  0570                         PUSHAL  T_WR_BLK_ERR
                    00  DD  0576                         PUSHL   #0
                    04  DD  0578                         PUSHL   #$ER
                            057A        ;;; TEST 12, SUBTEST 2, ERROR 4
          000100D0 9F    04    FB  057A                       CALLS    #$$M, @#DS$ERRHARD
                                   0581  7576         $DS_BNCOMPLETE  100$
                   03 50       E9  0581                       BLBC     R0, 100$
                                   0584  7577         $DS_ABORT       TEST
                         50    D4  0584                       CLRL     R0                   ; SEND A WARNING TO THE SUPERVISOR
                         04        0586                       RET                           ; TERMINATE TEST
                                   0587  7578 100$:   $DS_ABORT
          00010020 9F    6C    FA  0587                       CALLG    (AP), @#DS$ABORT
                                   058E  7579
               00000000'FF     94  058E  7580 110$:   CLRB     @A_BUFFER_BEGIN              ; clear buffer location with data
    FF1D 00000003'EF    01     03  9D  0594  7581     ACBB     #3,#1,B_COUNT,30$            ; br = not finished
                                   059E  7582                                               ; read end packet
          54    FFF3CB00 8F    D0  059E  7583 120$:   MOVL     #DELAY_TIME3,R4              ; load receive packet delay time
          55    00000000'FF    9E  05A5  7584         MOVAB    @A_BUFFER_BEGIN,R5           ; load buffer address to store data
              57    55    0E   C1  05AC  7585         ADDL3    #14,R5,R7                    ; load last expected address of data
               000020C8'EF     16  05B0  7586         JSB      MRECEIVE_TU58                ; go get data from TU58
          5A    0000068A'EF    9E  05B6  7587         MOVAB    MB_END_PACKET_4,R10          ; load address of expected end packet
          5B    00000000'FF    9E  05BD  7588         MOVAB    @A_BUFFER_BEGIN,R11          ; load address of actual end packet
                   57    55    D1  05C4  7589         CMPL     R5,R7                        ; check for correct number of bytes
                         31    13  05C7  7590         BEQL     150$                         ; br = correct number of bytes
          00000000'EF    08    88  05C9  7591         BISB2    #M_END_FLAG,B_FLAGS          ; set end error flag
                                   05D0  7592 122$:   $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=REC_ERR,-
                                   05D0  7593                           P1=#T_REC_END
               00001D2E'8F     DD  05D0                       PUSHL    #T_REC_END
               00001EB4'EF     DF  05D6                       PUSHAL   REC_ERR
               000018C8'EF     DF  05DC                       PUSHAL   T_WR_BLK_ERR
                         00    DD  05E2                       PUSHL    #0
                         05    DD  05E4                       PUSHL    #$ER
                                   05E6         ;;; TEST 12, SUBTEST 2, ERROR 5
          000100D0 9F    05    FB  05E6                       CALLS    #$$M, @#DS$ERRHARD
                                   05ED  7594         $DS_BNCOMPLETE  125$
                   03 50       E9  05ED                       BLBC     R0, 125$
                                   05F0  7595         $DS_ABORT       TEST
                         50    D4  05F0                       CLRL     R0                   ; SEND A WARNING TO THE SUPERVISOR
                         04        05F2                       RET                           ; TERMINATE TEST
                                   05F3  7596
                                   05F3  7597 125$:   $DS_ABORT
          00010020 9F    6C    FA  05F3                       CALLG    (AP), @#DS$ABORT
                                   05FA  7598                                               ; check end packet
                   5C    0E    D0  05FA  7599 150$:   MOVL     #14,R12                      ; load number of bytes to check
                                   05FD  7600         $DS_BREAK
          00010058 9F    6E    FA  05FD                       CALLG    (SP), @#DS$BREAK
               0000213B'EF     16  0604  7601         JSB      CHECK_PACKET                 ; go check end packet
                                   060A  7602         $DS_BNCOMPLETE  170$                  ; br = no error
                   24 50       E9  060A                       BLBC     R0, 170$
                                   060D  7603         $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=DATA_ERR_END
               00001E9D'EF     DF  060D                       PUSHAL   DATA_ERR_END
               000018C8'EF     DF  0613                       PUSHAL   T_WR_BLK_ERR
                         00    DD  0619                       PUSHL    #0
                         06    DD  061B                       PUSHL    #$ER
                                   061D         ;;; TEST 12, SUBTEST 2, ERROR 6
          000100D0 9F    04    FB  061D                       CALLS    #$$M, @#DS$ERRHARD
                                   0624  7604         $DS_BNCOMPLETE  160$
                   03 50       E9  0624                       BLBC     R0, 160$
                                   0627  7605         $DS_ABORT       TEST
```

```
                                        50   D4  0627                        CLRL    R0                ; SEND A WARNING TO THE SUPERVISOR
                                             04  0629                        RET                       ; TERMINATE TEST
                                                 062A   7606 160$:   $DS_ABORT
                   00010020 9F          6C   FA  062A                        CALLG   (AP), @#DS$ABORT
                                                 0631   7607 170$:
                                                 0631   7608         $DS_ENDSUB
                                                 0631        T12_S2_X:
         00010038 9F    000000FC'EF          FA  0631                        CALLG   ..$, @#DS$ENDSUB
                                                 063C   7609
                                                 063C   7610         $DS_PAGE
```

```
063C              .SBTTL  SUBTEST 12.3:   READ BLOCK
063C      ;+
063C      ; SUBTEST DESCRIPTION:
063C      ;
063C      ;       This subtest sends the TU58 a command packet to read the 512
063C      ;       bytes of data that were written in the previous subtest. A
063C      ;       timer is set which will time out if there is no response from the
063C      ;       TU58 between each byte transaction. The end packet and data packets
063C      ;       are checked for errors. If any errors are encountered they are
063C      ;       reported.
063C      ;
063C      ;
063C      ; SUBTEST STEPS:
063C      ;
063C      ;       SUBTEST READ BLOCK
063C      ;       :  Clear buffer
063C      ;       :  Point to read command packet
063C      ;       :  REPEAT
063C      ;       :  :  Send command packet byte at a time
063C      ;       :  :  Set timer
063C      ;       :  :  IF timer expired and no response
063C      ;       :  :  :  THEN
063C      ;       :  :  :  Report no response error
063C      ;       :  :  :  Exit test
063C      ;       :  :  ENDIF
063C      ;       :  : UNTIL all bytes sent
063C      ;       :  ENDREPEAT
063C      ;       :  REPEAT
063C      ;       :  :  Set timer
063C      ;       :  :  IF timer expired AND done set
063C      ;       :  :  :  THEN
063C      ;       :  :  :  :  Read byte
063C      ;       :  :  ENDIF
063C      ;       :  : UNTIL timer expired AND done clear
063C      ;       :  ENDREPEAT
063C      ;       :  IF expected byte count NEQ actual byte count
063C      ;       :  :  THEN
063C      ;       :  :  :  Report error
063C      ;       :  :  :  Exit test
063C      ;       :  ENDIF
063C      ;       :  REPEAT
063C      ;       :  :  Point to expected data packet
063C      ;       :  :  REPEAT
063C      ;       :  :  :  IF data expected NEQ data received
063C      ;       :  :  :  :  THEN
063C      ;       :  :  :  :  Report data error
063C      ;       :  :  :  :  Exit subtest
063C      ;       :  :  :  ENDIF
063C      ;       :  :  :  Point to next byte of data
063C      ;       :  :  : UNTIL all data in packet checked
063C      ;       :  :  ENDREPEAT
063C      ;       :  : UNTIL 4 data packets checked
063C      ;       :  ENDREPEAT
063C      ;       :  Point to expected end packet
063C      ;       :  REPEAT
```

```
                                     063C  7667 ;            :  :  IF data expected NEQ data received
                                     063C  7668 ;            :  :  :  THEN
                                     063C  7669 ;            :  :  :  Report data error
                                     063C  7670 ;            :  :  :  Exit subtest
                                     063C  7671 ;            :  :  ENDIF
                                     063C  7672 ;            :  : UNTIL all end packet bytes checked
                                     063C  7673 ;            :  ENDREPEAT
                                     063C  7674 ;            ENDSUBTEST READ BLOCK
                                     063C  7675 ;
                                     063C  7676 ;
                                     063C  7677 ;
                                     063C  7678 ;.page
                                     063C  7679 ;-
                                     063C  7680
                                     063C  7681              $DS_BGNSUB
                                     063C       T12_S3::
      00010030 9F   00000108'EF   FA 063C                           CALLG   $S$, @#DS$BGNSUB
                                     0647  7682                                                     ; clear buffers
                    00000000'EF   94 0647  7683              CLRB   B_FLAGS                         ; initialize flags
                58  00000000'FF   9E 064D  7684              MOVAB  @A_BUFFER_BEGIN,R8              ; load buffer starting address
                59  00000400 8F   D0 0654  7685              MOVL   #1024,R9                        ; do two pages
                    00002134'EF   16 065B  7686              JSB    CLEAR_BUFFER                    ; go clear buffers
                                     0661  7687                                                     ; send read command
                54  FFF3CB00 8F   D0 0661  7688              MOVL   #DELAY_TIME3,R4                 ; load send packet delay time
                55  00000350'EF   9E 0668  7689              MOVAB  MB_READ_2_CMD,R5                        ; point to read command packet
                         57   0E  D0 066F  7690              MOVL   #14,R7                          ; load transfer byte count
                    0000203D'EF   16 0672  7691              JSB    SEND_TU58                       ; send command packet
            24 00000000'EF    00  E1 0678  7692              BBC    #V_CLOCK_FLAG,B_FLAGS,2$ ; br = no timeout error
                                     0680  7693              $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=NO_READY
                    00001BEB'EF   DF 0680                           PUSHAL  NO_READY
                    000018EB'EF   DF 0686                           PUSHAL  T_RD_BLK_ERR
                              00  DD 068C                           PUSHL   #0
                              01  DD 068E                           PUSHL   #$ER
                                     0690       ;;; TEST 12, SUBTEST 3, ERROR 1
      000100D0 9F             04  FB 0690                           CALLS   #$$M, @#DS$ERRHARD
                                     0697  7694              $DS_BNCOMPLETE  1$
                         03 50    E9 0697                           BLBC    R0, 1$
                                     069A  7695              $DS_ABORT         TEST
                              50  D4 069A                           CLRL    R0                      ; SEND A WARNING TO THE SUPERVISOR
                              04     069C                           RET                             ; TERMINATE TEST
                                     069D  7696 1$:          $DS_ABORT
      00010020 9F             6C  FA 069D                           CALLG   (AP), @#DS$ABORT
                                     06A4  7697                                                     ; read data and end packets
                54  FFF3CB00 8F   D0 06A4  7698 2$:          MOVL   #DELAY_TIME3,R4                 ; load receive packet delay time
                55  00000000'FF   9E 06AB  7699              MOVAB  @A_BUFFER_BEGIN,R5              ; load buffer address to store data
           57   55  0000021E 8F   C1 06B2  7700              ADDL3  #542,R5,R7                      ; load last expected address of data
                    000020C8'EF   16 06BA  7701              JSB    MRECEIVE_TU58                   ; go get data from TU58
                5B  00000000'FF   9E 06C0  7702              MOVAB  @A_BUFFER_BEGIN,R11             ; load address of buffer
                5A  0000070A'EF   9E 06C7  7703              MOVAB  B_DATA_PKT_2,R10                ; load address of expected data
                         57   55  D1 06CE  7704              CMPL   R5,R7                           ; check for correct number of bytes
                              31  13 06D1  7705              BEQL   6$                              ; br = correct number of bytes
                00000000'EF   04  88 06D3  7706              BISB2  #M_DATA_FLAG,B_FLAGS            ; set data error flag
                                     06DA  7707 3$:          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=REC_ERR,-
                                     06DA  7708                               P1=#T_REC_DATA
                    00001D68'8F   DD 06DA                           PUSHL   #T_REC_DATA
```

```
          00001EB4'EF    DF  06E0                          PUSHAL  REC_ERR
          000018EB'EF    DF  06E6                          PUSHAL  T_RD_BLK_ERR
                   00    DD  06EC                          PUSHL   #0
                   02    DD  06EE                          PUSHL   #$ER
                             06F0            ;;; TEST 12, SUBTEST 3, ERROR 2
      000100D0 9F  05    FB  06F0                          CALLS   #$$M, @#DS$ERRHARD
                             06F7  7709            $DS_BNCOMPLETE  4$
                03 50    E9  06F7                          BLBC    R0, 4$
                             06FA  7710            $DS_ABORT       TEST
                   50    D4  06FA                          CLRL    R0                ; SEND A WARNING TO THE SUPERVISOR
                   04        06FC                          RET                       ; TERMINATE TEST
                             06FD  7711
                             06FD  7712 4$:        $DS_ABORT
      00010020 9F  6C    FA  06FD                          CALLG   (AP), @#DS$ABORT
                             0704  7713                                              ; check data packets
          00000006'EF    D4  0704  7714 6$:        CLRL    L_COUNT                   ; clear counter to check 4 packets
   5C     00000084 8F    D0  070A  7715            MOVL    #132,R12                  ; load number of bytes to check
                             0711  7716 7$:        $DS_BREAK
      00010058 9F  6E    FA  0711                          CALLG   (SP), @#DS$BREAK
          0000213B'EF    16  0718  7717            JSB     CHECK_PACKET              ; go check data packet
                             071E  7718            $DS_BNCOMPLETE  9$                ; br = no error
                24 50    E9  071E                          BLBC    R0, 9$
                             0721  7719            $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=DATA_ERR_DATA
          00001E75'EF    DF  0721                          PUSHAL  DATA_ERR_DATA
          000018EB'EF    DF  0727                          PUSHAL  T_RD_BLK_ERR
                   00    DD  072D                          PUSHL   #0
                   03    DD  072F                          PUSHL   #$ER
                             0731            ;;; TEST 12, SUBTEST 3, ERROR 3
      000100D0 9F  04    FB  0731                          CALLS   #$$M, @#DS$ERRHARD
                             0738  7720            $DS_BNCOMPLETE  8$
                03 50    E9  0738                          BLBC    R0, 8$
                             073B  7721            $DS_ABORT       TEST
                   50    D4  073B                          CLRL    R0                ; SEND A WARNING TO THE SUPERVISOR
                   04        073D                          RET                       ; TERMINATE TEST
                             073E  7722 8$:        $DS_ABORT
      00010020 9F  6C    FA  073E                          CALLG   (AP), @#DS$ABORT
      5B  00000084 8F    C0  0745  7723 9$:        ADDL2   #132,R11                  ; adjust pointer to next packet
   BD 00000006'EF  03    F3  074C  7724            AOBLEQ  #3,L_COUNT,7$             ; br = not finished checking data
                             0754  7725                                              ; check end packet
      5A  0000068A'EF    9E  0754  7726            MOVAB   MB_END_PACKET_4,R10       ; load address of expected end packet
              5C   0E    D0  075B  7727            MOVL    #14,R12                   ; load number of bytes to check
                             075E  7728            $DS_BREAK
      00010058 9F  6E    FA  075E                          CALLG   (SP), @#DS$BREAK
          0000213B'EF    16  0765  7729            JSB     CHECK_PACKET              ; go check end packet
                             076B  7730            $DS_BNCOMPLETE  11$               ; br = no error
                24 50    E9  076B                          BLBC    R0, 11$
                             076E  7731            $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=DATA_ERR_END
          00001E9D'EF    DF  076E                          PUSHAL  DATA_ERR_END
          000018EB'EF    DF  0774                          PUSHAL  T_RD_BLK_ERR
                   00    DD  077A                          PUSHL   #0
                   04    DD  077C                          PUSHL   #$ER
                             077E            ;;; TEST 12, SUBTEST 3, ERROR 4
      000100D0 9F  04    FB  077E                          CALLS   #$$M, @#DS$ERRHARD
                             0785  7732            $DS_BCOMPLETE   10$
                03 50    E8  0785                          BLBS    R0, 10$
```

```
                                     0788  7733         $DS_ABORT       TEST
                            50   D4  0788                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  078A                       RET                     ; TERMINATE TEST
                                     078B  7734 10$:    $DS_ABORT
              00010020 9F   6C   FA  078B                       CALLG   (AP), @#DS$ABORT
                                     0792  7735 11$:
                                     0792  7736
                                     0792  7737         $DS_ENDSUB
                                     0792       T12_S3_X:
     00010038 9F   00000108'EF   FA  0792                       CALLG   $S$, @#DS$ENDSUB
                                     079D  7738
                                     079D  7739         $DS_PAGE
```

```
079D            .SBTTL  SUBTEST 12.4:   WRITE MODIFIED
079D      ;+
079D      ; SUBTEST DESCRIPTION:
079D      ;
079D      ;         This subtest sends the TU58 a command packet to write modified
079D      ;         512 bytes of data. A timer is set which will time out if a
079D      ;         response is not received within the allotted time on all byte
079D      ;         transactions between the host and TU58. Four 128 byte data packets
079D      ;         are sent. Before each data packet is sent a continue is read from the
079D      ;         TU58 and when all data has been sent the end packet is read from the
079D      ;         TU58 and checked. If any errors are encountered they are reported.
079D      ;         This modified command will write all of the data and then back up and
079D      ;         check the checksum for data just written.
079D      ;
079D      ;
079D      ; SUBTEST STEPS:
079D      ;
079D      ;         SUBTEST WRITE MODIFIED
079D      ;         :  Clear buffer
079D      ;         :  Point to write command packet
079D      ;         :  REPEAT
079D      ;         :  :  Send command packet byte at a time
079D      ;         :  :  Set timer
079D      ;         :  :  IF timer expired AND no response
079D      ;         :  :  :  THEN
079D      ;         :  :  :  Report no response error
079D      ;         :  :  :  Exit test
079D      ;         :  :  ENDIF
079D      ;         :  : UNTIL all bytes sent
079D      ;         :  ENDREPEAT
079D      ;         :  DO for count = 1 to 4
079D      ;         :  :  Wait for continue
079D      ;         :  :  Set timer
079D      ;         :  :  IF timer expired AND no continue
079D      ;         :  :  :  THEN
079D      ;         :  :  :  Report no continue error
079D      ;         :  :  :  Exit test
079D      ;         :  :  ENDIF
079D      ;         :  :  Point to data packet
079D      ;         :  :  REPEAT
079D      ;         :  :  :  Send data packet byte at a time
079D      ;         :  :  :  Set timer
079D      ;         :  :  :  IF timer expired AND no response
079D      ;         :  :  :  :  THEN
079D      ;         :  :  :  :  Report no response error
079D      ;         :  :  :  :  Exit test
079D      ;         :  :  :  ENDIF
079D      ;         :  :  : UNTIL all bytes sent
079D      ;         :  :  ENDREPEAT
079D      ;         :  ENDDO
079D      ;         :  REPEAT
079D      ;         :  :  Set timer
079D      ;         :  :  IF timer expired AND done set
079D      ;         :  :  :  THEN
079D      ;         :  :  :  :  Read byte
```

```
                                079D  7796 ;            :  :  ENDIF
                                079D  7797 ;            :  : UNTIL timer expired AND done clear
                                079D  7798 ;            :  ENDREPEAT
                                079D  7799 ;            :  IF expected byte count NEQ actual byte count
                                079D  7800 ;            :  :  THEN
                                079D  7801 ;            :  :  :  Report error
                                079D  7802 ;            :  :  :  Exit test
                                079D  7803 ;            :  ENDIF
                                079D  7804 ;            :  Point to expected end packet
                                079D  7805 ;            :  REPEAT
                                079D  7806 ;            :  :  IF expected end packet data NEQ received end packet data
                                079D  7807 ;            :  :  :  THEN
                                079D  7808 ;            :  :  :  Report end packet data error
                                079D  7809 ;            :  :  ENDIF
                                079D  7810 ;            :  : UNTIL all data checked
                                079D  7811 ;            :  ENDREPEAT
                                079D  7812 ;            ENDSUBTEST WRITE MODIFIED
                                079D  7813 ;
                                079D  7814 ;
                                079D  7815 ;.page
                                079D  7816 ;-
                                079D  7817              $DS_BGNSUB
                                079D       T12_S4::
 00010030 9F   00000114'EF  FA  079D                            CALLG   $S$, a#DS$BGNSUB
                                07A8  7818                                                     ; clear buffers
               00000000'EF  94  07A8  7819              CLRB    B_FLAGS                        ; initialize flags
          58   00000000'FF  9E  07AE  7820              MOVAB   aA_BUFFER_BEGIN,R8             ; load buffer starting address
          59   00000400 8F  D0  07B5  7821              MOVL    #1024,R9                       ; do two pages
               00002134'EF  16  07BC  7822              JSB     CLEAR_BUFFER                   ; go clear buffers
                                07C2  7823                                                     ; send write command
          54   FFF3CB00 8F  D0  07C2  7824              MOVL    #DELAY_TIME3,R4                ; load send packet delay time
          55   0000037A'EF  9E  07C9  7825              MOVAB   MB_WRITE_4_CMD,R5              ; point to write command packet
                     57  0E D0  07D0  7826              MOVL    #14,R7                         ; load transfer byte count
               0000203D'EF  16  07D3  7827              JSB     SEND_TU58                      ; send command packet
      24 00000000'EF    00  E1  07D9  7828              BBC     #V_CLOCK_FLAG,B_FLAGS,2$ ; br = no timeout error
                                07E1  7829              $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_MOD_ERR,PRLINK=NO_READY
               00001BEB'EF  DF  07E1                            PUSHAL  NO_READY
               00001954'EF  DF  07E7                            PUSHAL  T_WR_MOD_ERR
                        00  DD  07ED                            PUSHL   #0
                        01  DD  07EF                            PUSHL   #$ER
                                07F1       ::: TEST 12, SUBTEST 4, ERROR 1
      000100D0 9F       04  FB  07F1                            CALLS   #$$M, a#DS$ERRHARD
                                07F8  7830              $DS_BNCOMPLETE  1$
                     03 50  E9  07F8                            BLBC    R0, 1$
                                07FB  7831              $DS_ABORT       TEST
                        50  D4  07FB                            CLRL    R0                     ; SEND A WARNING TO THE SUPERVISOR
                            04  07FD                            RET                            ; TERMINATE TEST
                                07FE  7832 1$:          $DS_ABORT
      00010020 9F       6C  FA  07FE                            CALLG   (AP), a#DS$ABORT
                                0805  7833                                                     ; read continue
               00000003'EF  94  0805  7834 2$:          CLRB    B_COUNT                        ; clear counter to send 4 data packets
          54   FFF3CB00 8F  D0  080B  7835 3$:          MOVL    #DELAY_TIME3,R4                ; load receive packet delay time
          55   00000000'FF  9E  0812  7836              MOVAB   aA_BUFFER_BEGIN,R5             ; load buffer address to store data
          57    55      01  C1  0819  7837              ADDL3   #1,R5,R7                       ; load last expected address of data
               000020C8'EF  16  081D  7838              JSB     MRECEIVE_TU58                  ; go get data from TU58
```

```
            5A   00000023'EF    9E  0823  7839          MOVAB   B_CONTINUE,R10              ; load address of expected data
            5B   00000000'FF    9E  082A  7840          MOVAB   @A_BUFFER_BEGIN,R11         ; load address of actual data
                      57  55    D1  0831  7841          CMPL    R5,R7                       ; check for correct number of byte
                          2A    13  0834  7842          BEQL    7$                          ; br = correct number of bytes
                                    0836  7843 4$:      $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_WR_MOD_ERR,PRLINK=REC_ERR,-
                                    0836  7844                            P1=#T_REC_CONT
                 00001D47'8F    DD  0836                         PUSHL    #T_REC_CONT
                 00001EB4'EF    DF  083C                         PUSHAL   REC_ERR
                 00001954'EF    DF  0842                         PUSHAL   T_WR_MOD_ERR
                          00    DD  0848                         PUSHL    #0
                          02    DD  084A                         PUSHL    #$ER
                                    084C            ;;; TEST 12, SUBTEST 4, ERROR 2
            000100D0 9F   05    FB  084C                         CALLS    #$$M, @#DS$ERRHARD
                                    0853  7845          $DS_BNCOMPLETE  5$
                      03  50    E9  0853                         BLBC     R0, 5$
                                    0856  7846          $DS_ABORT         TEST
                          50    D4  0856                         CLRL     R0                   ; SEND A WARNING TO THE SUPERVISOR
                          04    04  0858                         RET                           ; TERMINATE TEST
                                    0859  7847
                                    0859  7848 5$:      $DS_ABORT
            00010020 9F   6C    FA  0859                         CALLG    (AP), @#DS$ABORT
                                    0860  7849                                                 ; check continue
                      5C  01    D0  0860  7850 7$:      MOVL     #1,R12                      ; load number of bytes to check
                                    0863  7851          $DS_BREAK
            00010058 9F   6E    FA  0863                         CALLG    (SP), @#DS$BREAK
                 0000213B'EF    16  086A  7852          JSB      CHECK_PACKET                ; go check continue
                                    0870  7853          $DS_BNCOMPLETE  9$                   ; br = no error
                      24  50    E9  0870                         BLBC     R0, 9$
                                    0873  7854          $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_WR_MOD_ERR,PRLINK=DATA_ERR_CONT
                 00001E4D'EF    DF  0873                         PUSHAL   DATA_ERR_CONT
                 00001954'EF    DF  0879                         PUSHAL   T_WR_MOD_ERR
                          00    DD  087F                         PUSHL    #0
                          03    DD  0881                         PUSHL    #$ER
                                    0883            ;;; TEST 12, SUBTEST 4, ERROR 3
            000100D0 9F   04    FB  0883                         CALLS    #$$M, @#DS$ERRHARD
                                    088A  7855          $DS_BNCOMPLETE  8$
                      03  50    E9  088A                         BLBC     R0, 8$
                                    088D  7856          $DS_ABORT         TEST
                          50    D4  088D                         CLRL     R0                   ; SEND A WARNING TO THE SUPERVISOR
                          04        088F                         RET                           ; TERMINATE TEST
                                    0890  7857 8$:      $DS_ABORT
            00010020 9F   6C    FA  0890                         CALLG    (AP), @#DS$ABORT
                                    0897  7858                                                 ; send data packet
            54   FFFB6C20 8F    D0  0897  7859 9$:      MOVL     #DELAY_TIME1,R4             ; load send packet delay time
            55   0000070A'EF    9E  089E  7860          MOVAB    B_DATA_PKT_2,R5             ; point to data packet
            57   00000084 8F    D0  08A5  7861          MOVL     #132,R7                     ; load transfer byte count
                 0000203D'EF    16  08AC  7862          JSB      SEND_TU58                   ; send data packet
         24 00000000'EF   00    E1  08B2  7863          BBC      #V_CLOCK_FLAG,B_FLAGS,11$ ; br = no timeout error
                                    08BA  7864          $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_WR_MOD_ERR,PRLINK=NO_READY
                 00001BEB'EF    DF  08BA                         PUSHAL   NO_READY
                 00001954'EF    DF  08C0                         PUSHAL   T_WR_MOD_ERR
                          00    DD  08C6                         PUSHL    #0
                          04    DD  08C8                         PUSHL    #$ER
                                    08CA            ;;; TEST 12, SUBTEST 4, ERROR 4
            000100D0 9F   04    FB  08CA                         CALLS    #$$M, @#DS$ERRHARD
```

```
                                     08D1  7865         $DS_BNCOMPLETE   10$
                         03 50   E9  08D1                       BLBC     R0, 10$
                                     08D4  7866         $DS_ABORT        TEST
                            50   D4  08D4                       CLRL     R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  08D6                       RET                      ; TERMINATE TEST
                                     08D7  7867 10$:    $DS_ABORT
              00010020 9F   6C   FA  08D7                       CALLG    (AP), @#DS$ABORT
                   00000000'FF   94  08DE  7868 11$:    CLRB    @A_BUFFER_BEGIN          ; clear buffer location used for data
    FF1D 00000003'EF   01   03   9D  08E4  7869         ACBB    #3,#1,B_COUNT,3$         ; br = not finished
                                     08EE  7870                                          ; read end packet
              54   FFE17B80 8F   D0  08EE  7871         MOVL    #DELAY_TIMES,R4          ; load receive packet delay time
              55   00000000'FF   9E  08F5  7872         MOVAB   @A_BUFFER_BEGIN,R5       ; load buffer address to store data
                  57    55   0E   C1  08FC  7873         ADDL3   #14,R5,R7                ; load last expected address of data
                   000020C8'EF   16  0900  7874         JSB     MRECEIVE_TU58            ; go get data from TU58
              5A   0000068A'EF   9E  0906  7875         MOVAB   MB_END_PACKET_4,R10      ; load address of expected end packet
              5B   00000000'FF   9E  090D  7876         MOVAB   @A_BUFFER_BEGIN,R11      ; load address of actual end packet
                       57   55   D1  0914  7877         CMPL    R5,R7                    ; check for correct number of bytes
                            31   13  0917  7878         BEQL    15$                      ; br = correct number of bytes
              00000000'EF   08   88  0919  7879         BISB2   #M_END_FLAG,B_FLAGS      ; set end error flag
                                     0920  7880 12$:    $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_MOD_ERR,PRLINK=REC_ERR,-
                                     0920  7881                          P1=#T_REC_END
                   00001D2E'8F   DD  0920                       PUSHL    #T_REC_END
                   00001EB4'EF   DF  0926                       PUSHAL   REC_ERR
                   00001954'EF   DF  092C                       PUSHAL   T_WR_MOD_ERR
                            00   DD  0932                       PUSHL    #0
                            05   DD  0934                       PUSHL    #$ER
                                     0936       ;;; TEST 12, SUBTEST 4, ERROR 5
              000100D0 9F   05   FB  0936                       CALLS    #$$M, @#DS$ERRHARD
                                     093D  7882         $DS_BNCOMPLETE   13$
                         03 50   E9  093D                       BLBC     R0, 13$
                                     0940  7883         $DS_ABORT        TEST
                            50   D4  0940                       CLRL     R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  0942                       RET                      ; TERMINATE TEST
                                     0943  7884
                                     0943  7885 13$:    $DS_ABORT
              00010020 9F   6C   FA  0943                       CALLG    (AP), @#DS$ABORT
                                     094A  7886                                          ; check end packet
                       5C   0E   D0  094A  7887 15$:    MOVL    #14,R12                  ; load number of bytes to check
                                     094D  7888         $DS_BREAK
              00010058 9F   6E   FA  094D                       CALLG    (SP), @#DS$BREAK
                   0000213B'EF   16  0954  7889         JSB     CHECK_PACKET             ; go check end packet
                                     095A  7890         $DS_BNCOMPLETE   17$             ; br = no error
                         24 50   E9  095A                       BLBC     R0, 17$
                                     095D  7891         $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_MOD_ERR,PRLINK=DATA_ERR_END
                   00001E9D'EF   DF  095D                       PUSHAL   DATA_ERR_END
                   00001954'EF   DF  0963                       PUSHAL   T_WR_MOD_ERR
                            00   DD  0969                       PUSHL    #0
                            06   DD  096B                       PUSHL    #$ER
                                     096D       ;;; TEST 12, SUBTEST 4, ERROR 6
              000100D0 9F   04   FB  096D                       CALLS    #$$M, @#DS$ERRHARD
                                     0974  7892         $DS_BNCOMPLETE   16$
                         03 50   E9  0974                       BLBC     R0, 16$
                                     0977  7893         $DS_ABORT        TEST
                            50   D4  0977                       CLRL     R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  0979                       RET                      ; TERMINATE TEST
```

```
                                     097A  7894 16$:     $DS_ABORT
               00010020 9F    6C    FA   097A                    CALLG   (AP), a#DS$ABORT
                                     0981  7895 17$:
                                     0981  7896          $DS_ENDSUB
                                     0981       T12_S4_X:
        00010038 9F   00000114'EF   FA   0981                    CALLG   $S$, a#DS$ENDSUB
                                     098C  7897
                                     098C  7898          $DS_PAGE
```

```
098C        .SBTTL  SUBTEST 12.5:   READ MODIFIED
098C  ;+
098C  ; SUBTEST DESCRIPTION:
098C  ;
098C  ;     This subtest sends the TU58 a command packet to read modified
098C  ;     the 512 bytes of data written in the previous subtest. This
098C  ;     command will cause the TU58 to read the tape with an
098C  ;     increased threshold. A timer is set which will time out if there
098C  ;     is no response within the allotted time. After reading the first
098C  ;     byte of data, another timer is set which will time out if an end
098C  ;     packet is not received within the allotted time. If any data or
098C  ;     timer errors are encountered, they are reported.
098C  ;
098C  ;
098C  ; SUBTEST STEPS:
098C  ;
098C  ;     SUBTEST READ MODIFIED
098C  ;     :  Clear buffer
098C  ;     :  Point to read command packet
098C  ;     :  REPEAT
098C  ;     :  :  Send command packet byte at a time
098C  ;     :  :  Set timer
098C  ;     :  :  IF timer expired and no response
098C  ;     :  :  :  THEN
098C  ;     :  :  :  Report no response error
098C  ;     :  :  :  Exit test
098C  ;     :  :  ENDIF
098C  ;     :  : UNTIL all bytes sent
098C  ;     :  ENDREPEAT
098C  ;     :  REPEAT
098C  ;     :  :  Set timer
098C  ;     :  :  IF timer expired AND done set
098C  ;     :  :  :  THEN
098C  ;     :  :  :  :  Read byte
098C  ;     :  :  ENDIF
098C  ;     :  : UNTIL timer expired AND done clear
098C  ;     :  ENDREPEAT
098C  ;     :  IF expected byte count NEQ actual byte count
098C  ;     :  :  THEN
098C  ;     :  :  :  Report error
098C  ;     :  :  :  Exit test
098C  ;     :  ENDIF
098C  ;     :  REPEAT
098C  ;     :  :  Point to expected data packet
098C  ;     :  :  REPEAT
098C  ;     :  :  :  IF data expected NEQ data received
098C  ;     :  :  :  :  THEN
098C  ;     :  :  :  :  Report data error
098C  ;     :  :  :  :  Exit subtest
098C  ;     :  :  :  ENDIF
098C  ;     :  :  :  Point to next byte of data
098C  ;     :  :  : UNTIL all data in packet checked
098C  ;     :  :  ENDREPEAT
098C  ;     :  : UNTIL 4 data packets checked
098C  ;     :  ENDREPEAT
```

```
                                 098C  7955 ;         :  Point to expected end packet
                                 098C  7956 ;         :  REPEAT
                                 098C  7957 ;         :  :  IF end packet data expected NEQ end packet data received
                                 098C  7958 ;         :  :  :  THEN
                                 098C  7959 ;         :  :  :  Report end packet data error
                                 098C  7960 ;         :  :  :  Exit subtest
                                 098C  7961 ;         :  :  ENDIF
                                 098C  7962 ;         :  : UNTIL all end packet bytes checked
                                 098C  7963 ;         :  ENDREPEAT
                                 098C  7964 ;         ENDSUBTEST READ MODIFIED
                                 098C  7965 ;
                                 098C  7966 ;
                                 098C  7967 ;
                                 098C  7968 ;.page
                                 098C  7969 ;-
                                 098C  7970           $DS_BGNSUB
                                 098C       T12_S5::
       00010030 9F   00000120'EF   FA  098C                   CALLG    $S$, a#DS$BGNSUB
                                 0997  7971                                             ; clear buffers
                     00000000'EF   94  0997  7972    CLRB     B_FLAGS                   ; initialize flags
              58     00000000'FF   9E  099D  7973    MOVAB    aA_BUFFER_BEGIN,R8        ; load buffer starting address
              59     00000400 8F   D0  09A4  7974    MOVL     #1024,R9                  ; do two pages
                     00002134'EF   16  09AB  7975    JSB      CLEAR_BUFFER              ; go clear buffers
                                 09B1  7976                                             ; send read command
              54     FFF3CB00 8F   D0  09B1  7977    MOVL     #DELAY_TIME3,R4           ; load send packet delay time
              55     00000388'EF   9E  09B8  7978    MOVAB    MB_READ_4_CMD,R5                ; point to read command packet
                          57  0E   D0  09BF  7979    MOVL     #14,R7                    ; load transfer byte count
                     0000203D'EF   16  09C2  7980    JSB      SEND_TU58                 ; send command packet
          24 00000000'EF    00   E1  09C8  7981    BBC      #V_CLOCK_FLAG,B_FLAGS,2$ ; br = no timeout error
                                 09D0  7982           $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_RD_MOD_ERR,PRLINK=NO_READY
                     00001BEB'EF   DF  09D0                   PUSHAL   NO_READY
                     0000197A'EF   DF  09D6                   PUSHAL   T_RD_MOD_ERR
                              00   DD  09DC                   PUSHL    #0
                              01   DD  09DE                   PUSHL    #$ER
                                 09E0       ;;; TEST 12, SUBTEST 5, ERROR 1
            000100D0 9F    04   FB  09E0                   CALLS    #$$M, a#DS$ERRHARD
                                 09E7  7983           $DS_BNCOMPLETE  1$
                     03 50         E9  09E7                   BLBC     R0, 1$
                                 09EA  7984           $DS_ABORT       TEST
                              50   D4  09EA                   CLRL     R0             ; SEND A WARNING TO THE SUPERVISOR
                              04       09EC                   RET                     ; TERMINATE TEST
                                 09ED  7985 1$:       $DS_ABORT
            00010020 9F    6C   FA  09ED                   CALLG    (AP), a#DS$ABORT
                                 09F4  7986                                             ; read data and end packet
              54     FFF3CB00 8F   D0  09F4  7987 2$:    MOVL     #DELAY_TIME3,R4           ; load receive packet delay time
              55     00000000'FF   9E  09FB  7988    MOVAB    aA_BUFFER_BEGIN,R5        ; load buffer address to store data
          57  55     0000021E 8F   C1  0A02  7989    ADDL3    #542,R5,R7                ; load last expected address of data
                     000020C8'EF   16  0A0A  7990    JSB      MRECEIVE_TU58             ; go get data from TU58
              5B     00000000'FF   9E  0A10  7991    MOVAB    aA_BUFFER_BEGIN,R11       ; load address of actual data
              5A     0000070A'EF   9E  0A17  7992    MOVAB    B_DATA_PKT_2,R10          ; load address of expected data
                          57  55   D1  0A1E  7993    CMPL     R5,R7                     ; check for correct number of bytes
                          31  13       0A21  7994    BEQL     6$                        ; br = correct number of bytes
            00000000'EF    04   88  0A23  7995    BISB2    #M_DATA_FLAG,B_FLAGS      ; set data error flag
                                 0A2A  7996 3$:       $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_RD_MOD_ERR,PRLINK=REC_ERR,-
                                 0A2A  7997                              P1=#T_REC_DATA
```

```
                   00001D68'8F   DD  0A2A                           PUSHL   #T_REC_DATA
                   00001EB4'EF   DF  0A30                           PUSHAL  REC_ERR
                   0000197A'EF   DF  0A36                           PUSHAL  T_RD_MOD_ERR
                            00   DD  0A3C                           PUSHL   #0
                            02   DD  0A3E                           PUSHL   #$ER
                                     0A40           ;;; TEST 12, SUBTEST 5, ERROR 2
              000100D0 9F   05   FB  0A40                           CALLS   #$$M, @#DS$ERRHARD
                                     0A47  7998             $DS_BNCOMPLETE  4$
                         03 50   E9  0A47                           BLBC    R0, 4$
                                     0A4A  7999             $DS_ABORT       TEST
                            50   D4  0A4A                           CLRL    R0               ; SEND A WARNING TO THE SUPERVISOR
                            04   04  0A4C                           RET                      ; TERMINATE TEST
                                     0A4D  8000
                                     0A4D  8001 4$:         $DS_ABORT
              00010020 9F   6C   FA  0A4D                           CALLG   (AP), @#DS$ABORT
                                     0A54  8002                                              ; check data packets
                   00000006'EF   D4  0A54  8003 6$:         CLRL    L_COUNT                  ; clear counter to check 4 packets
              5C   00000084 8F   D0  0A5A  8004             MOVL    #132,R12                 ; load number of bytes to check
                                     0A61  8005 7$:         $DS_BREAK
              00010058 9F   6E   FA  0A61                           CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16  0A68  8006             JSB     CHECK_PACKET             ; go check data packet
                                     0A6E  8007             $DS_BNCOMPLETE  9$               ; br = no error
                         24 50   E9  0A6E                           BLBC    R0, 9$
                                     0A71  8008             $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_MOD_ERR,PRLINK=DATA_ERR_DATA
                   00001E75'EF   DF  0A71                           PUSHAL  DATA_ERR_DATA
                   0000197A'EF   DF  0A77                           PUSHAL  T_RD_MOD_ERR
                            00   DD  0A7D                           PUSHL   #0
                            03   DD  0A7F                           PUSHL   #$ER
                                     0A81           ;;; TEST 12, SUBTEST 5, ERROR 3
              000100D0 9F   04   FB  0A81                           CALLS   #$$M, @#DS$ERRHARD
                                     0A88  8009             $DS_BNCOMPLETE  8$
                         03 50   E9  0A88                           BLBC    R0, 8$
                                     0A8B  8010             $DS_ABORT       TEST
                            50   D4  0A8B                           CLRL    R0               ; SEND A WARNING TO THE SUPERVISOR
                            04       0A8D                           RET                      ; TERMINATE TEST
                                     0A8E  8011 8$:         $DS_ABORT
              00010020 9F   6C   FA  0A8E                           CALLG   (AP), @#DS$ABORT
              5B   00000084 8F   C0  0A95  8012 9$:         ADDL2   #132,R11                 ; adjust pointer to next packet
           BD 00000006'EF   03   F3  0A9C  8013             AOBLEQ  #3,L_COUNT,7$            ; br = not finished checking data
                                     0AA4  8014                                              ; check end packet
              5A   0000068A'EF   9E  0AA4  8015             MOVAB   MB_END_PACKET_4,R10      ; load address of expected end packet
                        5C  0E   D0  0AAB  8016             MOVL    #14,R12                  ; load number of bytes to check
                                     0AAE  8017             $DS_BREAK
              00010058 9F   6E   FA  0AAE                           CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16  0AB5  8018             JSB     CHECK_PACKET             ; go check end packet
                                     0ABB  8019             $DS_BNCOMPLETE  11$              ; br = no error
                         24 50   E9  0ABB                           BLBC    R0, 11$
                                     0ABE  8020             $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_MOD_ERR,PRLINK=DATA_ERR_END
                   00001E9D'EF   DF  0ABE                           PUSHAL  DATA_ERR_END
                   0000197A'EF   DF  0AC4                           PUSHAL  T_RD_MOD_ERR
                            00   DD  0ACA                           PUSHL   #0
                            04   DD  0ACC                           PUSHL   #$ER
                                     0ACE           ;;; TEST 12, SUBTEST 5, ERROR 4
              000100D0 9F   04   FB  0ACE                           CALLS   #$$M, @#DS$ERRHARD
                                     0AD5  8021             $DS_BNCOMPLETE  10$
```

```
                      03 50   E9  0AD5                      BLBC    R0, 10$
                                  0AD8  8022         $DS_ABORT      TEST
                         50   D4  0AD8                      CLRL    R0                 ; SEND A WARNING TO THE SUPERVISOR
                              04  0ADA                      RET                        ; TERMINATE TEST
                                  0ADB  8023 10$:    $DS_ABORT
             00010020 9F   6C  FA 0ADB                      CALLG   (AP), @#DS$ABORT
                                  0AE2  8024 11$:    $DS_ENDSUB
                                  0AE2       T12_S5_X:
      00010038 9F   00000120'EF  FA 0AE2                    CALLG   $S$, @#DS$ENDSUB
                                  0AED  8025         $DS_PAGE
```

```
0AED             .SBTTL  SUBTEST 12.6:   WRITE RECORD
0AED  8028 ;+
0AED  8029 ; SUBTEST DESCRIPTION:
0AED  8030 ;
0AED  8031 ;        This subtest sends the TU58 a command packet to write one record
0AED  8032 ;        with 128 bytes of data. A continue flag is returned and the
0AED  8033 ;        data packet is sent. All byte transactions between the host and
0AED  8034 ;        TU58 are timed and will timeout if there is no response within
0AED  8035 ;        the allotted time. If a timer expires while expecting a data
0AED  8036 ;        transfer to occur, an error is reported. The data packet
0AED  8037 ;        contains alternating bytes of all one's and all zero's. Both
0AED  8038 ;        end and data packets are checked and if any errors have occured,
0AED  8039 ;        they are reported.
0AED  8040 ;
0AED  8041 ;
0AED  8042 ; SUBTEST STEPS:
0AED  8043 ;
0AED  8044 ;        SUBTEST WRITE RECORD
0AED  8045 ;        :  Clear buffer
0AED  8046 ;        :  Point to write command packet
0AED  8047 ;        :  REPEAT
0AED  8048 ;        :  :  Send command packet byte at a time
0AED  8049 ;        :  :  Set timer
0AED  8050 ;        :  :  IF timer expired AND no response
0AED  8051 ;        :  :  :  THEN
0AED  8052 ;        :  :  :  Report no response error
0AED  8053 ;        :  :  :  Exit test
0AED  8054 ;        :  :  ENDIF
0AED  8055 ;        :  : UNTIL all bytes sent
0AED  8056 ;        :  ENDREPEAT
0AED  8057 ;        :  Wait for continue
0AED  8058 ;        :  Set timer
0AED  8059 ;        :  IF timer expired AND no continue
0AED  8060 ;        :  :  THEN
0AED  8061 ;        :  :  Report no continue error
0AED  8062 ;        :  :  Exit test
0AED  8063 ;        :  ENDIF
0AED  8064 ;        :  Point to data packet
0AED  8065 ;        :  REPEAT
0AED  8066 ;        :  :  Send data packet byte at a time
0AED  8067 ;        :  :  Set timer
0AED  8068 ;        :  :  IF timer expired AND no response
0AED  8069 ;        :  :  :  THEN
0AED  8070 ;        :  :  :  Report no response error
0AED  8071 ;        :  :  :  Exit test
0AED  8072 ;        :  :  ENDIF
0AED  8073 ;        :  : UNTIL all bytes sent
0AED  8074 ;        :  ENDREPEAT
0AED  8075 ;        :  REPEAT
0AED  8076 ;        :  :  Set timer
0AED  8077 ;        :  :  IF timer expired AND done set
0AED  8078 ;        :  :  :  THEN
0AED  8079 ;        :  :  :  :  Read byte
0AED  8080 ;        :  :  ENDIF
0AED  8081 ;        :  : UNTIL timer expired AND done clear
```

```
                                    0AED  8082 ;         :  ENDREPEAT
                                    0AED  8083 ;         :  IF expected byte count NEQ actual byte count
                                    0AED  8084 ;         :  :  THEN
                                    0AED  8085 ;         :  :  :  Report error
                                    0AED  8086 ;         :  :  :  Exit test
                                    0AED  8087 ;         :  ENDIF
                                    0AED  8088 ;         :  REPEAT
                                    0AED  8089 ;         :  :  Point to expected end packet
                                    0AED  8090 ;         :  :  IF expected end packet data NEQ received end packet data
                                    0AED  8091 ;         :  :  :  THEN
                                    0AED  8092 ;         :  :  :  Report end packet data error
                                    0AED  8093 ;         :  :  ENDIF
                                    0AED  8094 ;         :  : UNTIL all data checked
                                    0AED  8095 ;         :  ENDREPEAT
                                    0AED  8096 ;        ENDSUBTEST WRITE RECORD
                                    0AED  8097 ;
                                    0AED  8098 ;
                                    0AED  8099 ;
                                    0AED  8100 ;.page
                                    0AED  8101 ;-
                                    0AED  8102
                                    0AED  8103          $DS_BGNSUB
                                    0AED       T12_S6::
00010030 9F    0000012C'EF     FA   0AED                        CALLG   $S$, @#DS$BGNSUB
               00000000'EF     94   0AF8  8104          CLRB    B_FLAGS                        ; initialize flags
               00000002'EF     94   0AFE  8105          CLRB    B_FLAGS3
                                    0B04  8106                                                 ; clear buffers
      58       00000000'FF     9E   0B04  8107          MOVAB   @A_BUFFER_BEGIN,R8             ; load buffer starting address
      59       00000400 8F     D0   0B0B  8108          MOVL    #1024,R9                       ; do two pages
               00002134'EF     16   0B12  8109          JSB     CLEAR_BUFFER                   ; go clear buffers
                                    0B18  8110                                                 ; send write command
      54       FFE17B80 8F     D0   0B18  8111          MOVL    #DELAY_TIME5,R4                ; load send packet delay time
      55       0000035E'EF     9E   0B1F  8112          MOVAB   MB_WRITE_3_CMD,R5              ; point to write command packet
                    57   0E    D0   0B26  8113          MOVL    #14,R7                         ; load transfer byte count
               0000203D'EF     16   0B29  8114          JSB     SEND_TU58                      ; send command packet
         24 00000000'EF   00   E1   0B2F  8115          BBC     #V_CLOCK_FLAG,B_FLAGS,20$ ; br = no timeout error
                                    0B37  8116          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=NO_READY
               00001BEB'EF     DF   0B37                        PUSHAL  NO_READY
               0000190D'EF     DF   0B3D                        PUSHAL  T_WR_REC_ERR
                         00    DD   0B43                        PUSHL   #0
                         01    DD   0B45                        PUSHL   #$ER
                                    0B47       ;;; TEST 12, SUBTEST 6, ERROR 1
     000100D0 9F         04    FB   0B47                        CALLS   #$$M, @#DS$ERRHARD
                                    0B4E  8117          $DS_BNCOMPLETE  10$
                      03 50    E9   0B4E                        BLBC    R0, 10$
                                    0B51  8118          $DS_ABORT       TEST
                         50    D4   0B51                        CLRL    R0                     ; SEND A WARNING TO THE SUPERVISOR
                         04    04   0B53                        RET                            ; TERMINATE TEST
                                    0B54  8119 10$:     $DS_ABORT
     00010020 9F         6C    FA   0B54                        CALLG   (AP), @#DS$ABORT
                                    0B5B  8120                                                 ; read continue
      54       FFF3CB00 8F     D0   0B5B  8121 20$:     MOVL    #DELAY_TIME3,R4                ; load receive packet delay time
      55       00000000'FF     9E   0B62  8122          MOVAB   @A_BUFFER_BEGIN,R5             ; load buffer address to store data
          57    55       01    C1   0B69  8123          ADDL3   #1,R5,R7                       ; load last expected address of data
               000020C8'EF     16   0B6D  8124          JSB     MRECEIVE_TU58                  ; go get data from TU58
```

```
            5A    00000023'EF    9E  0B73  8125          MOVAB    B_CONTINUE,R10             ; load address of expected data
            5B    00000000'FF    9E  0B7A  8126          MOVAB    @A_BUFFER_BEGIN,R11        ; load address of actual data
                        57   55  D1  0B81  8127          CMPL     R5,R7                      ; check for correct number of bytes
                             2A  13  0B84  8128          BEQL     60$                        ; br = correct number of bytes
                                     0B86  8129 25$:     $DS_ERRHARD_S      ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=REC_ERR,-
                                     0B86  8130                             P1=#T_REC_CONT
                  00001D47'8F    DD  0B86                         PUSHL   #T_REC_CONT
                  00001EB4'EF    DF  0B8C                         PUSHAL  REC_ERR
                  0000190D'EF    DF  0B92                         PUSHAL  T_WR_REC_ERR
                             00  DD  0B98                         PUSHL   #0
                             02  DD  0B9A                         PUSHL   #$ER
                                     0B9C            ;;; TEST 12, SUBTEST 6, ERROR 2
            000100D0 9F      05  FB  0B9C                         CALLS   #$$M, @#DS$ERRHARD
                                     0BA3  8131          $DS_BNCOMPLETE  30$
                          03 50  E9  0BA3                         BLBC    R0, 30$
                                     0BA6  8132          $DS_ABORT       TEST
                             50  D4  0BA6                         CLRL    R0                 ; SEND A WARNING TO THE SUPERVISOR
                             04  0BA8                             RET                        ; TERMINATE TEST
                                     0BA9  8133
                                     0BA9  8134 30$:     $DS_ABORT
            00010020 9F      6C  FA  0BA9                         CALLG   (AP), @#DS$ABORT
                                     0BB0  8135                                              ; check continue
                        5C   01  D0  0BB0  8136 60$:     MOVL     #1,R12                     ; load number of bytes to check
                                     0BB3  8137          $DS_BREAK
            00010058 9F      6E  FA  0BB3                         CALLG   (SP), @#DS$BREAK
                  0000213B'EF    16  0BBA  8138          JSB      CHECK_PACKET               ; go check end packet
                                     0BC0  8139          $DS_BNCOMPLETE  80$                 ; br = no error
                          24 50  E9  0BC0                         BLBC    R0, 80$
                                     0BC3  8140          $DS_ERRHARD_S      ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=DATA_ERR_CONT
                  00001E4D'EF    DF  0BC3                         PUSHAL  DATA_ERR_CONT
                  0000190D'EF    DF  0BC9                         PUSHAL  T_WR_REC_ERR
                             00  DD  0BCF                         PUSHL   #0
                             03  DD  0BD1                         PUSHL   #$ER
                                     0BD3            ;;; TEST 12, SUBTEST 6, ERROR 3
            000100D0 9F      04  FB  0BD3                         CALLS   #$$M, @#DS$ERRHARD
                                     0BDA  8141          $DS_BNCOMPLETE  70$
                          03 50  E9  0BDA                         BLBC    R0, 70$
                                     0BDD  8142          $DS_ABORT       TEST
                             50  D4  0BDD                         CLRL    R0                 ; SEND A WARNING TO THE SUPERVISOR
                             04  0BDF                             RET                        ; TERMINATE TEST
                                     0BE0  8143 70$:     $DS_ABORT
            00010020 9F      6C  FA  0BE0                         CALLG   (AP), @#DS$ABORT
                                     0BE7  8144                                              ; send data packet
            54    FFFB6C20 8F    D0  0BE7  8145 80$:     MOVL     #DELAY_TIME1,R4            ; load send packet delay time
            55    0000070A'EF    9E  0BEE  8146          MOVAB    B_DATA_PKT_2,R5            ; point to data packet
            57    00000084 8F    D0  0BF5  8147          MOVL     #132,R7                    ; load transfer byte count
                  0000203D'EF    16  0BFC  8148          JSB      SEND_TU58                  ; send data packet
         24 00000000'EF      00  E1  0C02  8149          BBC      #V_CLOCK_FLAG,B_FLAGS,100$ ; br = no timeout error
                                     0C0A  8150          $DS_ERRHARD_S      ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=NO_READY
                  00001BEB'EF    DF  0C0A                         PUSHAL  NO_READY
                  0000190D'EF    DF  0C10                         PUSHAL  T_WR_REC_ERR
                             00  DD  0C16                         PUSHL   #0
                             04  DD  0C18                         PUSHL   #$ER
                                     0C1A            ;;; TEST 12, SUBTEST 6, ERROR 4
            000100D0 9F      04  FB  0C1A                         CALLS   #$$M, @#DS$ERRHARD
```

```
                                   0C21  8151        $DS_BNCOMPLETE 90$
                        03 50  E9  0C21                     BLBC    R0, 90$
                                   0C24  8152        $DS_ABORT      TEST
                           50  D4  0C24                     CLRL    R0             ; SEND A WARNING TO THE SUPERVISOR
                               04  0C26                     RET                    ; TERMINATE TEST
                                   0C27  8153 90$:   $DS_ABORT
             00010020 9F   6C  FA  0C27                     CALLG   (AP), @#DS$ABORT
                                   0C2E  8154                                      ; read end packet
                  00000002'EF  96  0C2E  8155 100$:  INCB   B_FLAGS3               ; set data done flag
             54   FFC2F700 8F  D0  0C34  8156        MOVL   #DELAY_TIME6,R4        ; load receive packet delay time
             55   00000000'FF  9E  0C3B  8157        MOVAB  @A_BUFFER_BEGIN,R5     ; load address to store actual data
                 57   55   0E  C1  0C42  8158        ADDL3  #14,R5,R7              ; load last expected address of data
                  000020C8'EF  16  0C46  8159        JSB    MRECEIVE_TU58          ; go get data from TU58
             5A   00000698'EF  9E  0C4C  8160        MOVAB  MB_END_PACKET_5,R10    ; load address of expected end packet
             5B   00000000'FF  9E  0C53  8161        MOVAB  @A_BUFFER_BEGIN,R11    ; load address of actual end packet
                      57   55  D1  0C5A  8162        CMPL   R5,R7                  ; check for correct number of bytes
                           31  13  0C5D  8163        BEQL   140$                   ; br = correct number of bytes
             00000000'EF   08  88  0C5F  8164        BISB2  #M_END_FLAG,B_FLAGS    ; set end error flag
                                   0C66  8165 105$:  $DS_ERRHARD_S  ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=REC_ERR,-
                                   0C66  8166                       P1=#T_REC_END
                  00001D2E'8F  DD  0C66                     PUSHL   #T_REC_END
                  00001EB4'EF  DF  0C6C                     PUSHAL  REC_ERR
                  0000190D'EF  DF  0C72                     PUSHAL  T_WR_REC_ERR
                           00  DD  0C78                     PUSHL   #0
                           05  DD  0C7A                     PUSHL   #$ER
                                   0C7C       ;;; TEST 12, SUBTEST 6, ERROR 5
             000100D0 9F   05  FB  0C7C                     CALLS   #$$M, @#DS$ERRHARD
                                   0C83  8167        $DS_BNCOMPLETE 110$
                        03 50  E9  0C83                     BLBC    R0, 110$
                                   0C86  8168        $DS_ABORT      TEST
                           50  D4  0C86                     CLRL    R0             ; SEND A WARNING TO THE SUPERVISOR
                               04  0C88                     RET                    ; TERMINATE TEST
                                   0C89  8169
                                   0C89  8170 110$:  $DS_ABORT
             00010020 9F   6C  FA  0C89                     CALLG   (AP), @#DS$ABORT
                                   0C90  8171                                      ; check end packet
                      5C   0E  D0  0C90  8172 140$:  MOVL   #14,R12                ; load number of bytes to check
                                   0C93  8173        $DS_BREAK
             00010058 9F   6E  FA  0C93                     CALLG   (SP), @#DS$BREAK
                  0000213B'EF  16  0C9A  8174        JSB    CHECK_PACKET           ; go check end packet
                                   0CA0  8175        $DS_BNCOMPLETE 160$           ; br = no error
                        24 50  E9  0CA0                     BLBC    R0, 160$
                                   0CA3  8176        $DS_ERRHARD_S  ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=DATA_ERR_END
                  00001E9D'EF  DF  0CA3                     PUSHAL  DATA_ERR_END
                  0000190D'EF  DF  0CA9                     PUSHAL  T_WR_REC_ERR
                           00  DD  0CAF                     PUSHL   #0
                           06  DD  0CB1                     PUSHL   #$ER
                                   0CB3       ;;; TEST 12, SUBTEST 6, ERROR 6
             000100D0 9F   04  FB  0CB3                     CALLS   #$$M, @#DS$ERRHARD
                                   0CBA  8177        $DS_BNCOMPLETE 150$
                        03 50  E9  0CBA                     BLBC    R0, 150$
                                   0CBD  8178        $DS_ABORT      TEST
                           50  D4  0CBD                     CLRL    R0             ; SEND A WARNING TO THE SUPERVISOR
                               04  0CBF                     RET                    ; TERMINATE TEST
                                   0CC0  8179 150$:  $DS_ABORT
```

```
              00010020 9F    6C    FA  0CC0                      CALLG   (AP), a#DS$ABORT
                                       0CC7  8180 160$:
                                       0CC7  8181         $DS_ENDSUB
                                       0CC7       T12_S6_X:
     00010038 9F    0000012C'EF    FA  0CC7                      CALLG   $S$, a#DS$ENDSUB
                                       0CD2  8182
                                       0CD2  8183         $DS_PAGE
```

```
0CD2                .SBTTL  SUBTEST 12.7:   READ RECORD
0CD2        ;+
0CD2        ; SUBTEST DESCRIPTION:
0CD2        ;
0CD2        ;       This subtest sends the TU58 a command packet to read the 128
0CD2        ;       bytes of data that were written in the previous subtest. A
0CD2        ;       timer is set and times out if there is no response between all
0CD2        ;       byte transactions. The end packet and data packet are checked for
0CD2        ;       errors. If any errors are encountered they are reported.
0CD2        ;
0CD2        ;
0CD2        ; SUBTEST STEPS:
0CD2        ;
0CD2        ;       SUBTEST READ RECORD
0CD2        ;       :  Clear buffer
0CD2        ;       :  Point to read command packet
0CD2        ;       :  REPEAT
0CD2        ;       :  :  Send command packet byte at a time
0CD2        ;       :  :  Set timer
0CD2        ;       :  :  IF timer expired and no response
0CD2        ;       :  :  :  THEN
0CD2        ;       :  :  :  Report no response error
0CD2        ;       :  :  :  Exit test
0CD2        ;       :  :  ENDIF
0CD2        ;       :  : UNTIL all bytes sent
0CD2        ;       :  ENDREPEAT
0CD2        ;       :  REPEAT
0CD2        ;       :  :  Set timer
0CD2        ;       :  :  IF timer expired AND done set
0CD2        ;       :  :  :  THEN
0CD2        ;       :  :  :  :  Read byte
0CD2        ;       :  :  ENDIF
0CD2        ;       :  : UNTIL timer expired AND done clear
0CD2        ;       :  ENDREPEAT
0CD2        ;       :  IF expected byte count NEQ actual byte count
0CD2        ;       :  :  THEN
0CD2        ;       :  :  :  Report error
0CD2        ;       :  :  :  Exit test
0CD2        ;       :  ENDIF
0CD2        ;       :  Point to expected data packet
0CD2        ;       :  REPEAT
0CD2        ;       :  :  IF data expected NEQ data received
0CD2        ;       :  :  :  THEN
0CD2        ;       :  :  :  Report data error
0CD2        ;       :  :  :  Exit subtest
0CD2        ;       :  :  ENDIF
0CD2        ;       :  :  Point to next byte of data
0CD2        ;       :  : UNTIL all data in packet checked
0CD2        ;       :  ENDREPEAT
0CD2        ;       :  Point to expected end packet
0CD2        ;       :  REPEAT
0CD2        ;       :  :  IF expected end packet data NEQ received end packet data
0CD2        ;       :  :  :  THEN
0CD2        ;       :  :  :  Report end packet data error
0CD2        ;       :  :  :  Exit subtest
```

```
                                        0CD2  8240 ;        :  :  ENDIF
                                        0CD2  8241 ;        :  : UNTIL all end packet bytes checked
                                        0CD2  8242 ;        :  ENDREPEAT
                                        0CD2  8243 ;        ENDSUBTEST READ RECORD
                                        0CD2  8244 ;
                                        0CD2  8245 ;
                                        0CD2  8246 ;
                                        0CD2  8247 ;.page
                                        0CD2  8248 ;-
                                        0CD2  8249          $DS_BGNSUB
                                        0CD2       T12_S7::
       00010030 9F    00000138'EF   FA  0CD2                        CALLG   $S$, a#DS$BGNSUB
                                        0CDD  8250                                                      ; clear buffers
                      00000000'EF   94  0CDD  8251          CLRB    B_FLAGS                             ; initialize flags
                 58   00000000'FF   9E  0CE3  8252          MOVAB   aA_BUFFER_BEGIN,R8                  ; load buffer starting address
                 59   00000400 8F   D0  0CEA  8253          MOVL    #1024,R9                            ; do two pages
                      00002134'EF   16  0CF1  8254          JSB     CLEAR_BUFFER                        ; go clear buffers
                                        0CF7  8255                                                      ; send read command
                 54   FFF3CB00 8F   D0  0CF7  8256          MOVL    #DELAY_TIME3,R4                     ; load send packet delay time
                 55   0000036C'EF   9E  0CFE  8257          MOVAB   MB_READ_3_CMD,R5                            ; point to read command packet
                            57  0E  D0  0D05  8258          MOVL    #14,R7                              ; load transfer byte count
                      0000203D'EF   16  0D08  8259          JSB     SEND_TU58                           ; send command packet
              24 00000000'EF    00  E1  0D0E  8260          BBC     #V_CLOCK_FLAG,B_FLAGS,2$ ; br = no timeout error
                                        0D16  8261          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=NO_READY
                      00001BEB'EF   DF  0D16                        PUSHAL  NO_READY
                      00001931'EF   DF  0D1C                        PUSHAL  T_RD_REC_ERR
                                00  DD  0D22                        PUSHL   #0
                                01  DD  0D24                        PUSHL   #$ER
                                        0D26       ;;; TEST 12, SUBTEST 7, ERROR 1
            000100D0 9F         04  FB  0D26                        CALLS   #$$M, a#DS$ERRHARD
                                        0D2D  8262          $DS_BNCOMPLETE  1$
                             03 50  E9  0D2D                        BLBC    R0, 1$
                                        0D30  8263          $DS_ABORT       TEST
                                50  D4  0D30                        CLRL    R0                          ; SEND A WARNING TO THE SUPERVISOR
                                    04  0D32                        RET                                 ; TERMINATE TEST
                                        0D33  8264 1$:      $DS_ABORT
            00010020 9F         6C  FA  0D33                        CALLG   (AP), a#DS$ABORT
                                        0D3A  8265                                                      ; read data and end packet
                 54   FFF3CB00 8F   D0  0D3A  8266 2$:      MOVL    #DELAY_TIME3,R4                     ; load receive packet delay time
                 55   00000000'FF   9E  0D41  8267          MOVAB   aA_BUFFER_BEGIN,R5                  ; load buffer address to store data
           57    55   00000092 8F   C1  0D48  8268          ADDL3   #146,R5,R7                          ; load last expected address of data
                      000020C8'EF   16  0D50  8269          JSB     MRECEIVE_TU58                       ; go get data from TU58
                 5B   00000000'FF   9E  0D56  8270          MOVAB   aA_BUFFER_BEGIN,R11                 ; load address of actual data
                 5A   0000070A'EF   9E  0D5D  8271          MOVAB   B_DATA_PKT_2,R10                    ; load address of expected data
                            57  55  D1  0D64  8272          CMPL    R5,R7                               ; check for correct number of bytes
                            31  13  0D67  8273              BEQL    6$                                  ; br = correct number of bytes
            00000000'EF     04  88  0D69  8274          BISB2   #M_DATA_FLAG,B_FLAGS                ; set data error flag
                                        0D70  8275 3$:      $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=REC_ERR,-
                                        0D70  8276                           P1=#T_REC_DATA
                      00001D68'8F   DD  0D70                        PUSHL   #T_REC_DATA
                      00001EB4'EF   DF  0D76                        PUSHAL  REC_ERR
                      00001931'EF   DF  0D7C                        PUSHAL  T_RD_REC_ERR
                                00  DD  0D82                        PUSHL   #0
                                02  DD  0D84                        PUSHL   #$ER
                                        0D86       ;;; TEST 12, SUBTEST 7, ERROR 2
```

```
        000100D0 9F    05   FB  0D86                      CALLS   #$$M, @#DS$ERRHARD
                                0D8D  8277          $DS_BNCOMPLETE  4$
                  03 50     E9  0D8D                      BLBC    R0, 4$
                                0D90  8278          $DS_ABORT       TEST
                     50     D4  0D90                      CLRL    R0                    ; SEND A WARNING TO THE SUPERVISOR
                            04  0D92                      RET                           ; TERMINATE TEST
                                0D93  8279
                                0D93  8280 4$:      $DS_ABORT
        00010020 9F    6C   FA  0D93                      CALLG   (AP), @#DS$ABORT
                                0D9A  8281                                              ; check data packet
        5C    00000084 8F   D0  0D9A  8282 6$:      MOVL    #132,R12                    ; load number of bytes to check
                                0DA1  8283          $DS_BREAK
        00010058 9F    6E   FA  0DA1                      CALLG   (SP), @#DS$BREAK
              0000213B'EF   16  0DA8  8284          JSB     CHECK_PACKET                ; go check data packet
                                0DAE  8285          $DS_BNCOMPLETE  8$                  ; br = no error
                  24 50     E9  0DAE                      BLBC    R0, 8$
                                0DB1  8286          $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=DATA_ERR_DATA
              00001E75'EF   DF  0DB1                      PUSHAL  DATA_ERR_DATA
              00001931'EF   DF  0DB7                      PUSHAL  T_RD_REC_ERR
                     00     DD  0DBD                      PUSHL   #0
                     03     DD  0DBF                      PUSHL   #$ER
                                0DC1            ;;; TEST 12, SUBTEST 7, ERROR 3
        000100D0 9F    04   FB  0DC1                      CALLS   #$$M, @#DS$ERRHARD
                                0DC8  8287          $DS_BNCOMPLETE  7$
                  03 50     E9  0DC8                      BLBC    R0, 7$
                                0DCB  8288          $DS_ABORT       TEST
                     50     D4  0DCB                      CLRL    R0                    ; SEND A WARNING TO THE SUPERVISOR
                            04  0DCD                      RET                           ; TERMINATE TEST
                                0DCE  8289 7$:      $DS_ABORT
        00010020 9F    6C   FA  0DCE                      CALLG   (AP), @#DS$ABORT
                                0DD5  8290                                              ; check end packet
        5B    00000084 8F   C0  0DD5  8291 8$:      ADDL2   #132,R11                    ; adjust pointer to next packet
        5A    00000698'EF   9E  0DDC  8292          MOVAB   MB_END_PACKET_5,R10         ; load address of expected end packet
                  5C 0E     D0  0DE3  8293          MOVL    #14,R12                     ; load number of bytes to check
                                0DE6  8294          $DS_BREAK
        00010058 9F    6E   FA  0DE6                      CALLG   (SP), @#DS$BREAK
              0000213B'EF   16  0DED  8295          JSB     CHECK_PACKET                ; go check end packet
                                0DF3  8296          $DS_BNCOMPLETE  10$                 ; br = no error
                  24 50     E9  0DF3                      BLBC    R0, 10$
                                0DF6  8297          $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=DATA_ERR_END
              00001E9D'EF   DF  0DF6                      PUSHAL  DATA_ERR_END
              00001931'EF   DF  0DFC                      PUSHAL  T_RD_REC_ERR
                     00     DD  0E02                      PUSHL   #0
                     04     DD  0E04                      PUSHL   #$ER
                                0E06            ;;; TEST 12, SUBTEST 7, ERROR 4
        000100D0 9F    04   FB  0E06                      CALLS   #$$M, @#DS$ERRHARD
                                0E0D  8298          $DS_BNCOMPLETE  9$                  ; br = error, but continue
                  03 50     E9  0E0D                      BLBC    R0, 9$
                                0E10  8299          $DS_ABORT       TEST
                     50     D4  0E10                      CLRL    R0                    ; SEND A WARNING TO THE SUPERVISOR
                            04  0E12                      RET                           ; TERMINATE TEST
                                0E13  8300 9$:      $DS_ABORT
        00010020 9F    6C   FA  0E13                      CALLG   (AP), @#DS$ABORT
                                0E1A  8301 10$:
                                0E1A  8302          $DS_ENDSUB
```

```
                                  0E1A        T12_S7_X:
    00010038 9F    00000138'EF    FA  0E1A                    CALLG    $S$, @#DS$ENDSUB
                                  0E25  8303
                                  0E25  8304        $DS_PAGE
```

```
0E25             .SBTTL  SUBTEST 12.8:   WRITE BLOCK - PADDED ZEROS
0E25    ;+
0E25    ; SUBTEST DESCRIPTION:
0E25    ;
0E25    ;        This subtest sends the TU58 a series of command packets to check
0E25    ;        that a block written with less than 512 bytes of data, will
0E25    ;        pad the remainder of the block with data bytes of zeros. This
0E25    ;        is accomplished by writing two consecutive blocks each with
0E25    ;        512 bytes of all ones. Then a command packet is sent to
0E25    ;        write the first block with one byte of alternating ones and
0E25    ;        zeroes. This command should write the first byte with
0E25    ;        10101010 and the remaining 511 bytes with 00000000. The second
0E25    ;        block should still contain 512 bytes of all one's data. This data
0E25    ;        is read and checked. All end packets and data packets are checked as
0E25    ;        the subtest progresses. All byte transactions between host and TU58 are
0E25    ;        timed. If any errors are encountered, they are reported.
0E25    ;
0E25    ;
0E25    ; SUBTEST STEPS:
0E25    ;
0E25    ;
0E25    ;        SUBTEST WRITE BLOCK PADDED ZEROS
0E25    ;        :  Clear buffer
0E25    ;        :  Point to first write command packet
0E25    ;        :  DO for count = 1 to 3
0E25    ;        :  :  REPEAT
0E25    ;        :  :  :  Send command packet byte at a time
0E25    ;        :  :  :  Set timer
0E25    ;        :  :  :  IF timer expired AND no response
0E25    ;        :  :  :  :  THEN
0E25    ;        :  :  :  :  Report no response error
0E25    ;        :  :  :  :  Exit test
0E25    ;        :  :  :  ENDIF
0E25    ;        :  :  : UNTIL all bytes sent
0E25    ;        :  :  ENDREPEAT
0E25    ;        :  :  DO for count = 1 to 4
0E25    ;        :  :  :  Wait for continue
0E25    ;        :  :  :  Set timer
0E25    ;        :  :  :  IF timer expired AND no continue
0E25    ;        :  :  :  :  THEN
0E25    ;        :  :  :  :  Report no continue error
0E25    ;        :  :  :  :  Exit test
0E25    ;        :  :  :  ENDIF
0E25    ;        :  :  :  Point to data packet
0E25    ;        :  :  :  REPEAT
0E25    ;        :  :  :  :  Send data packet byte at a time
0E25    ;        :  :  :  :  Set timer
0E25    ;        :  :  :  :  IF timer expired AND no response
0E25    ;        :  :  :  :  :  THEN
0E25    ;        :  :  :  :  :  Report no response error
0E25    ;        :  :  :  :  :  Exit test
0E25    ;        :  :  :  :  ENDIF
0E25    ;        :  :  :  : UNTIL all bytes sent
0E25    ;        :  :  :  ENDREPEAT
0E25    ;        :  :  ENDDO
```

```
0E25  8361 ;       :  :  REPEAT
0E25  8362 ;       :  :  :  Set timer
0E25  8363 ;       :  :  :  IF timer expired AND done set
0E25  8364 ;       :  :  :  :  THEN
0E25  8365 ;       :  :  :  :  :  Read byte
0E25  8366 ;       :  :  :  ENDIF
0E25  8367 ;       :  :  : UNTIL timer expired AND done clear
0E25  8368 ;       :  :  ENDREPEAT
0E25  8369 ;       :  :  IF expected byte count NEQ actual byte count
0E25  8370 ;       :  :  :  THEN
0E25  8371 ;       :  :  :  :  Report error
0E25  8372 ;       :  :  :  :  Exit test
0E25  8373 ;       :  :  ENDIF
0E25  8374 ;       :  :  REPEAT
0E25  8375 ;       :  :  :  Point to expected end packet
0E25  8376 ;       :  :  :  IF expected end data NEQ received end data
0E25  8377 ;       :  :  :  :  THEN
0E25  8378 ;       :  :  :  :  Report end packet data error
0E25  8379 ;       :  :  :  ENDIF
0E25  8380 ;       :  :  : UNTIL all data checked
0E25  8381 ;       :  :  ENDREPEAT
0E25  8382 ;       :  :  Point to next write command
0E25  8383 ;       :  ENDDO
0E25  8384 ;       :  Clear buffer
0E25  8385 ;       :  Point to first read command packet
0E25  8386 ;       :  DO for count = 1 to 2
0E25  8387 ;       :  :  REPEAT
0E25  8388 ;       :  :  :  Send command packet byte at a time
0E25  8389 ;       :  :  :  Set timer
0E25  8390 ;       :  :  :  IF timer expired and no response
0E25  8391 ;       :  :  :  :  THEN
0E25  8392 ;       :  :  :  :  Report no response error
0E25  8393 ;       :  :  :  :  Exit test
0E25  8394 ;       :  :  :  ENDIF
0E25  8395 ;       :  :  : UNTIL all bytes sent
0E25  8396 ;       :  :  ENDREPEAT
0E25  8397 ;       :  :  :  REPEAT
0E25  8398 ;       :  :  :  Set timer
0E25  8399 ;       :  :  :  IF timer expired AND done set
0E25  8400 ;       :  :  :  :  THEN
0E25  8401 ;       :  :  :  :  :  Read byte
0E25  8402 ;       :  :  :  ENDIF
0E25  8403 ;       :  :  : UNTIL timer expired AND done clear
0E25  8404 ;       :  :  ENDREPEAT
0E25  8405 ;       :  :  IF expected byte count NEQ actual byte count
0E25  8406 ;       :  :  :  THEN
0E25  8407 ;       :  :  :  :  Report error
0E25  8408 ;       :  :  :  :  Exit test
0E25  8409 ;       :  :  ENDIF
0E25  8410 ;       :  :  REPEAT
0E25  8411 ;       :  :  :  Point to expected data packet
0E25  8412 ;       :  :  :  REPEAT
0E25  8413 ;       :  :  :  : IF data expected NEQ data received
0E25  8414 ;       :  :  :  :  :  THEN
0E25  8415 ;       :  :  :  :  :  Report data error
```

```
                                     0E25  8416 ;       :  :  :  :  :  Exit subtest
                                     0E25  8417 ;       :  :  :  :  ENDIF
                                     0E25  8418 ;       :  :  :  :  Point to next byte of data
                                     0E25  8419 ;       :  :  :  : UNTIL all data in packet checked
                                     0E25  8420 ;       :  :  :  ENDREPEAT
                                     0E25  8421 ;       :  :  : UNTIL 4 data packets checked
                                     0E25  8422 ;       :  :  ENDREPEAT
                                     0E25  8423 ;       :  :  Point to expected end packet
                                     0E25  8424 ;       :  :  REPEAT
                                     0E25  8425 ;       :  :  :  IF end data expected NEQ end data received
                                     0E25  8426 ;       :  :  :  :  THEN
                                     0E25  8427 ;       :  :  :  :  Report end packet data error
                                     0E25  8428 ;       :  :  :  :  Exit subtest
                                     0E25  8429 ;       :  :  :  ENDIF
                                     0E25  8430 ;       :  :  : UNTIL all end packet bytes checked
                                     0E25  8431 ;       :  :  ENDREPEAT
                                     0E25  8432 ;       :  ENDDO
                                     0E25  8433 ;       ENDSUBTEST WRITE BLOCK PADDED ZEROS
                                     0E25  8434 ;
                                     0E25  8435 ;
                                     0E25  8436 ;
                                     0E25  8437 ;.page
                                     0E25  8438 ;-
                                     0E25  8439         $DS_BGNSUB
                                     0E25       T12_S8::
     00010030 9F   00000144'EF   FA  0E25                       CALLG   $S$, @#DS$BGNSUB
                   00000004'EF   94  0E30  8440         CLRB    B_COUNT1                ; initialize command counter
              00000005'EF   03   90  0E36  8441         MOVB    #3,B_COUNT2             ; init data packet counter
     0000000E'EF   00000084 8F   D0  0E3D  8442         MOVL    #132,L_COUNT3           ; init data counter
              53   000015B3'EF   9E  0E48  8443         MOVAB   MCOMMAND_TBL_1,R3       ; load pointer to command table
                                     0E4F  8444                                         ; clear buffers
                   00000000'EF   94  0E4F  8445 10$:    CLRB    B_FLAGS                 ; initialize flags
              58   00000000'FF   9E  0E55  8446         MOVAB   @A_BUFFER_BEGIN,R8      ; load buffer starting address
              59   00000600 8F   D0  0E5C  8447         MOVL    #BUFFER_SIZE,R9         ; do all pages
                   00002134'EF   16  0E63  8448         JSB     CLEAR_BUFFER            ; go clear buffers
                                     0E69  8449                                         ; send write command
              54   FFF3CB00 8F   D0  0E69  8450         MOVL    #DELAY_TIME3,R4         ; load send packet delay time
                      55    93   DE  0E70  8451         MOVAL   @(R3)+,R5               ; point to write command packet
                      57    0E   D0  0E73  8452         MOVL    #14,R7                  ; load transfer byte count
                   0000203D'EF   16  0E76  8453         JSB     SEND_TU58               ; send command packet
           24 00000000'EF   00   E1  0E7C  8454         BBC     #V_CLOCK_FLAG,B_FLAGS,30$ ; br = no timeout error
                                     0E84  8455         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=NO_READY
                   00001BEB'EF   DF  0E84                       PUSHAL  NO_READY
                   000018C8'EF   DF  0E8A                       PUSHAL  T_WR_BLK_ERR
                            00   DD  0E90                       PUSHL   #0
                            01   DD  0E92                       PUSHL   #$ER
                                     0E94       ;;; TEST 12, SUBTEST 8, ERROR 1
              000100D0 9F   04   FB  0E94                       CALLS   #$$M, @#DS$ERRHARD
                                     0E9B  8456         $DS_BNCOMPLETE  20$
                         03 50   E9  0E9B                       BLBC    R0, 20$
                                     0E9E  8457         $DS_ABORT       TEST
                            50   D4  0E9E                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                            04       0EA0                       RET                     ; TERMINATE TEST
                                     0EA1  8458 20$:    $DS_ABORT
              00010020 9F   6C   FA  0EA1                       CALLG   (AP), @#DS$ABORT
```

```
                                     0EA8  8459                                        ; read continue
                   00000003'EF   94  0EA8  8460 30$:   CLRB    B_COUNT                 ; initialize data packet counter
              54   FFC2F700 8F   D0  0EAE  8461 35$:   MOVL    #DELAY_TIME6,R4         ; load receive packet delay time
              55   00000000'FF   9E  0EB5  8462        MOVAB   @A_BUFFER_BEGIN,R5      ; load buffer address to store data
                  57    55    01   C1  0EBC  8463        ADDL3   #1,R5,R7                ; load last expected address of data
                   000020C8'EF   16  0EC0  8464        JSB     MRECEIVE_TU58           ; go get data from TU58
              5A   00000023'EF   9E  0EC6  8465        MOVAB   B_CONTINUE,R10          ; load address of expected data
              5B   00000000'FF   9E  0ECD  8466        MOVAB   @A_BUFFER_BEGIN,R11     ; load address of actual data
                       57    55   D1  0ED4  8467        CMPL    R5,R7                   ; check for correct number of bytes
                            2A   13  0ED7  8468        BEQL    70$                     ; br = correct number of bytes
                                     0ED9  8469 37$:   $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=REC_ERR,-
                                     0ED9  8470                        P1=#T_REC_CONT
                   00001D47'8F   DD  0ED9                      PUSHL   #T_REC_CONT
                   00001EB4'EF   DF  0EDF                      PUSHAL  REC_ERR
                   000018C8'EF   DF  0EE5                      PUSHAL  T_WR_BLK_ERR
                            00   DD  0EEB                      PUSHL   #0
                            02   DD  0EED                      PUSHL   #$ER
                                     0EEF       ;;; TEST 12, SUBTEST 8, ERROR 2
              000100D0 9F   05   FB  0EEF                      CALLS   #$$M, @#DS$ERRHARD
                                     0EF6  8471        $DS_BNCOMPLETE  40$
                         03 50   E9  0EF6                      BLBC    R0, 40$
                                     0EF9  8472        $DS_ABORT       TEST
                            50   D4  0EF9                      CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  0EFB                      RET                     ; TERMINATE TEST
                                     0EFC  8473
                                     0EFC  8474 40$:   $DS_ABORT
              00010020 9F   6C   FA  0EFC                      CALLG   (AP), @#DS$ABORT
                                     0F03  8475                                        ; check continue
                       5C   01   D0  0F03  8476 70$:   MOVL    #1,R12                  ; load number of bytes to check
                                     0F06  8477        $DS_BREAK
              00010058 9F   6E   FA  0F06                      CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16  0F0D  8478        JSB     CHECK_PACKET            ; go check continue
                                     0F13  8479        $DS_BNCOMPLETE  90$             ; br = no error
                         24 50   E9  0F13                      BLBC    R0, 90$
                                     0F16  8480        $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=DATA ERR_CONT
                   00001E4D'EF   DF  0F16                      PUSHAL  DATA_ERR_CONT
                   000018C8'EF   DF  0F1C                      PUSHAL  T_WR_BLK_ERR
                            00   DD  0F22                      PUSHL   #0
                            03   DD  0F24                      PUSHL   #$ER
                                     0F26       ;;; TEST 12, SUBTEST 8, ERROR 3
              000100D0 9F   04   FB  0F26                      CALLS   #$$M, @#DS$ERRHARD
                                     0F2D  8481        $DS_BNCOMPLETE  80$
                         03 50   E9  0F2D                      BLBC    R0, 80$
                                     0F30  8482        $DS_ABORT       TEST
                            50   D4  0F30                      CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  0F32                      RET                     ; TERMINATE TEST
                                     0F33  8483 80$:   $DS_ABORT
              00010020 9F   6C   FA  0F33                      CALLG   (AP), @#DS$ABORT
                                     0F3A  8484                                        ; send data packet
              54   FFFB6C20 8F   D0  0F3A  8485 90$:   MOVL    #DELAY_TIME1,R4         ; load send packet delay time
                       55   93   DE  0F41  8486        MOVAL   @(R3)+,R5               ; point to data packet
              57   0000000E'EF   D0  0F44  8487        MOVL    L_COUNT3,R7             ; load transfer byte count
                   0000203D'EF   16  0F4B  8488        JSB     SEND_TU58               ; send data packet
           24 00000000'EF   00   E1  0F51  8489        BBC     #V_CLOCK_FLAG,B_FLAGS,110$ ; br = no timeout error
                                     0F59  8490        $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=NO_READY
                   00001BEB'EF   DF  0F59                       PUSHAL  NO_READY
                   000018C8'EF   DF  0F5F                       PUSHAL  T_WR_BLK_ERR
                            00   DD  0F65                       PUSHL   #0
                            04   DD  0F67                       PUSHL   #$ER
                                     0F69       ;;; TEST 12, SUBTEST 8, ERROR 4
              000100D0 9F   04   FB  0F69                       CALLS   #$$M, @#DS$ERRHARD
                                     0F70  8491         $DS_BNCOMPLETE  100$
                         03 50   E9  0F70                       BLBC    R0, 100$
                                     0F73  8492         $DS_ABORT       TEST
                            50   D4  0F73                       CLRL    R0               ; SEND A WARNING TO THE SUPERVISOR
                            04   04  0F75                       RET                      ; TERMINATE TEST
                                     0F76  8493 100$:   $DS_ABORT
              00010020 9F   6C   FA  0F76                       CALLG   (AP), @#DS$ABORT
                                     0F7D  8494
                   00000000'FF   94  0F7D  8495 110$:   CLRB    @A_BUFFER_BEGIN          ; clear buffer location used for data
00000003'EF   01   00000005'EF   9D  0F83  8496         ACBB    B_COUNT2,#1,B_COUNT,35$ ; br = not finished sending data
                               FF1D  0F8F
                                     0F91  8497                                          ; read end packet
              54   FFE17B80 8F   D0  0F91  8498 115$:   MOVL    #DELAY_TIMES,R4          ; load receive packet delay time
              55   00000000'FF   9E  0F98  8499         MOVAB   @A_BUFFER_BEGIN,R5       ; load buffer address to store data
                  57   55   0E   C1  0F9F  8500         ADDL3   #14,R5,R7                ; load last expected address of data
                   000020C8'EF   16  0FA3  8501         JSB     MRECEIVE_TU58            ; go get data from TU58
                       5A   93   DE  0FA9  8502         MOVAL   @(R3)+,R10               ; load address of expected end packet
              5B   00000000'FF   9E  0FAC  8503         MOVAB   @A_BUFFER_BEGIN,R11      ; load address of actual end packet
                       57   55   D1  0FB3  8504         CMPL    R5,R7                    ; check for correct number of bytes
                            31   13  0FB6  8505         BEQL    150$                     ; br = correct number of bytes
              00000000'EF   08   88  0FB8  8506         BISB2   #M_END_FLAG,B_FLAGS      ; set end error flag
                                     0FBF  8507 117$:   $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=REC_ERR,-
                                     0FBF  8508                         P1=#T_REC_END
                   00001D2E'8F   DD  0FBF                       PUSHL   #T_REC_END
                   00001EB4'EF   DF  0FC5                       PUSHAL  REC_ERR
                   000018C8'EF   DF  0FCB                       PUSHAL  T_WR_BLK_ERR
                            00   DD  0FD1                       PUSHL   #0
                            05   DD  0FD3                       PUSHL   #$ER
                                     0FD5       ;;; TEST 12, SUBTEST 8, ERROR 5
              000100D0 9F   05   FB  0FD5                       CALLS   #$$M, @#DS$ERRHARD
                                     0FDC  8509         $DS_BNCOMPLETE  120$
                         03 50   E9  0FDC                       BLBC    R0, 120$
                                     0FDF  8510         $DS_ABORT       TEST
                            50   D4  0FDF                       CLRL    R0               ; SEND A WARNING TO THE SUPERVISOR
                                 04  0FE1                       RET                      ; TERMINATE TEST
                                     0FE2  8511
                                     0FE2  8512 120$:   $DS_ABORT
              00010020 9F   6C   FA  0FE2                       CALLG   (AP), @#DS$ABORT
                                     0FE9  8513                                          ; check end packet
                       5C   0E   D0  0FE9  8514 150$:   MOVL    #14,R12                  ; load number of bytes to check
                                     0FEC  8515         $DS_BREAK
              00010058 9F   6E   FA  0FEC                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16  0FF3  8516         JSB     CHECK_PACKET             ; go check end packet
                                     0FF9  8517         $DS_BNCOMPLETE  160$             ; br = no error
                         24 50   E9  0FF9                       BLBC    R0, 160$
                                     0FFC  8518         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=DATA_ERR_END
                   00001E9D'EF   DF  0FFC                       PUSHAL  DATA_ERR_END
                   000018C8'EF   DF  1002                       PUSHAL  T_WR_BLK_ERR
                            00   DD  1008                       PUSHL   #0
                            06   DD  100A                       PUSHL   #$ER
                                     100C       ;;; TEST 12, SUBTEST 8, ERROR 6
              000100D0 9F    04   FB  100C                       CALLS   #$$M, @#DS$ERRHARD
                                     1013  8519         $DS_BNCOMPLETE  155$
                         03 50   E9  1013                       BLBC    R0, 155$
                                     1016  8520         $DS_ABORT       TEST
                            50   D4  1016                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  1018                       RET                     ; TERMINATE TEST
                                     1019  8521 155$:   $DS_ABORT
              00010020 9F    6C   FA  1019                       CALLG   (AP), @#DS$ABORT
                1A 00000005'EF   E9  1020  8522 160$:   BLBC    B_COUNT2,165$           ; go do read commands if writes done
    FE1E 00000004'EF   01   01   9D  1027  8523         ACBB    #1,#1,B_COUNT1,10$      ; br = not finished
                   00000005'EF   94  1031  8524         CLRB    B_COUNT2                ; set data packet counter for 1 packet
              0000000E'EF   05   D0  1037  8525         MOVL    #5,L_COUNT3             ; set data count to 5 bytes
                          FE0E   31  103E  8526         BRW     10$                     ; go do last write
                   00000004'EF   94  1041  8527 165$:   CLRB    B_COUNT1                ; initialize command counter
                                     1047  8528                                         ; clear buffers
                   00000000'EF   94  1047  8529 170$:   CLRB    B_FLAGS                 ; initialize flags
              58   00000000'FF   9E  104D  8530         MOVAB   @A_BUFFER_BEGIN,R8      ; load buffer starting address
              59   00000600 8F   D0  1054  8531         MOVL    #BUFFER_SIZE,R9         ; do all pages
                   00002134'EF   16  105B  8532         JSB     CLEAR_BUFFER            ; go clear buffers
                                     1061  8533                                         ; send read command
              54   FFFB6C20 8F   D0  1061  8534         MOVL    #DELAY_TIME1,R4         ; load send packet delay time
                       55   93   DE  1068  8535         MOVAL   @(R3)+,R5               ; point to read command packet
                       57   0E   D0  106B  8536         MOVL    #14,R7                  ; load transfer byte count
                   0000203D'EF   16  106E  8537         JSB     SEND_TU58               ; send command packet
           24 00000000'EF   00   E1  1074  8538         BBC     #V_CLOCK_FLAG,B_FLAGS,190$ ; br = no timeout error
                                     107C  8539         $DS_ERRHARD_S  ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=NO_READY
                   00001BEB'EF   DF  107C                       PUSHAL  NO_READY
                   000018EB'EF   DF  1082                       PUSHAL  T_RD_BLK_ERR
                            00   DD  1088                       PUSHL   #0
                            07   DD  108A                       PUSHL   #$ER
                                     108C       ;;; TEST 12, SUBTEST 8, ERROR 7
              000100D0 9F    04   FB  108C                       CALLS   #$$M, @#DS$ERRHARD
                                     1093  8540         $DS_BNCOMPLETE  180$
                         03 50   E9  1093                       BLBC    R0, 180$
                                     1096  8541         $DS_ABORT       TEST
                            50   D4  1096                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  1098                       RET                     ; TERMINATE TEST
                                     1099  8542 180$:   $DS_ABORT
              00010020 9F    6C   FA  1099                       CALLG   (AP), @#DS$ABORT
                                     10A0  8543                                         ; read data and end packets
              54   FFE17B80 8F   D0  10A0  8544 190$:   MOVL    #DELAY_TIME5,R4         ; load receive packet delay time
              55   00000000'FF   9E  10A7  8545         MOVAB   @A_BUFFER_BEGIN,R5      ; load buffer address to store data
         57   55   0000021E 8F   C1  10AE  8546         ADDL3   #542,R5,R7              ; load last expected address of data
                   000020C8'EF   16  10B6  8547         JSB     MRECEIVE_TU58           ; go get data from TU58
              5B   00000000'FF   9E  10BC  8548         MOVAB   @A_BUFFER_BEGIN,R11     ; load address of actual data
                       5A   93   DE  10C3  8549         MOVAL   @(R3)+,R10              ; load address of expected data
                       57   55   D1  10C6  8550         CMPL    R5,R7                   ; check for correct number of bytes
                            31   13  10C9  8551         BEQL    230$                    ; br = correct number of bytes
              00000000'EF   04   88  10CB  8552         BISB2   #M_DATA_FLAG,B_FLAGS    ; set data error flag
                                     10D2  8553 195$:   $DS_ERRHARD_S  ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=REC_ERR,-
                                     10D2  8554                         P1=#T_REC_DATA
                   00001D68'8F   DD  10D2                       PUSHL   #T_REC_DATA
                   00001EB4'EF   DF  10D8                       PUSHAL  REC_ERR
                   000018EB'EF   DF   10DE                       PUSHAL  T_RD_BLK_ERR
                            00   DD   10E4                       PUSHL   #0
                            08   DD   10E6                       PUSHL   #$ER
                                      10E8       ;;; TEST 12, SUBTEST 8, ERROR 8
              000100D0 9F    05   FB   10E8                       CALLS   #$$M, @#DS$ERRHARD
                                      10EF  8555         $DS_BNCOMPLETE  200$
                         03 50   E9   10EF                       BLBC    R0, 200$
                                      10F2  8556         $DS_ABORT       TEST
                            50   D4   10F2                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04   10F4                       RET                     ; TERMINATE TEST
                                      10F5  8557
                                      10F5  8558 200$:   $DS_ABORT
              00010020 9F    6C   FA   10F5                       CALLG   (AP), @#DS$ABORT
                                      10FC  8559                                         ; check data packets
                   00000003'EF   94   10FC  8560 230$:   CLRB    B_COUNT                 ; clear counter to check 4 packets
              5C   00000084 8F   D0   1102  8561         MOVL    #132,R12                ; load number of bytes to check
                                      1109  8562 240$:   $DS_BREAK
              00010058 9F    6E   FA   1109                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16   1110  8563         JSB     CHECK_PACKET            ; go check data packet
                                      1116  8564         $DS_BNCOMPLETE  260$            ; br = no error
                         24 50   E9   1116                       BLBC    R0, 260$
                                      1119  8565         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=DATA_ERR_DATA
                   00001E75'EF   DF   1119                       PUSHAL  DATA_ERR_DATA
                   000018EB'EF   DF   111F                       PUSHAL  T_RD_BLK_ERR
                            00   DD   1125                       PUSHL   #0
                            09   DD   1127                       PUSHL   #$ER
                                      1129       ;;; TEST 12, SUBTEST 8, ERROR 9
              000100D0 9F    04   FB   1129                       CALLS   #$$M, @#DS$ERRHARD
                                      1130  8566         $DS_BNCOMPLETE  250$
                         03 50   E9   1130                       BLBC    R0, 250$
                                      1133  8567         $DS_ABORT       TEST
                            50   D4   1133                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04   1135                       RET                     ; TERMINATE TEST
                                      1136  8568 250$:   $DS_ABORT
              00010020 9F    6C   FA   1136                       CALLG   (AP), @#DS$ABORT
              5B   00000084 8F   C0   113D  8569 260$:   ADDL2   #132,R11                ; adjust pointer to next packet
                       5A    93   DE   1144  8570         MOVAL   @(R3)+,R10              ; load address of expected data
    FFB8 00000003'EF   01    03   9D   1147  8571         ACBB    #3,#1,B_COUNT,240$      ; br = not finished checking data
                                      1151  8572                                         ; check end packet
                       5C    0E   D0   1151  8573         MOVL    #14,R12                 ; load number of bytes to check
                                      1154  8574         $DS_BREAK
              00010058 9F    6E   FA   1154                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16   115B  8575         JSB     CHECK_PACKET            ; go check end packet
                                      1161  8576         $DS_BNCOMPLETE  270$            ; br = no error
                         24 50   E9   1161                       BLBC    R0, 270$
                                      1164  8577         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=DATA_ERR_END
                   00001E9D'EF   DF   1164                       PUSHAL  DATA_ERR_END
                   000018EB'EF   DF   116A                       PUSHAL  T_RD_BLK_ERR
                            00   DD   1170                       PUSHL   #0
                            0A   DD   1172                       PUSHL   #$ER
                                      1174       ;;; TEST 12, SUBTEST 8, ERROR 10
              000100D0 9F    04   FB   1174                       CALLS   #$$M, @#DS$ERRHARD
                                      117B  8578         $DS_BNCOMPLETE  265$
                         03 50   E9   117B                       BLBC    R0, 265$
                                      117E  8579         $DS_ABORT       TEST
                            50   D4  117E                       CLRL    R0               ; SEND A WARNING TO THE SUPERVISOR
                                 04  1180                       RET                      ; TERMINATE TEST
                                     1181  8580 265$:   $DS_ABORT
              00010020 9F   6C   FA  1181                       CALLG   (AP), @#DS$ABORT
    FEB5 00000003'EF   01   01   9D  1188  8581 270$:   ACBB    #1,#1,B_COUNT,170$       ; br = not finished reading
                                     1192  8582
                                     1192  8583
                                     1192  8584
                                     1192  8585
                                     1192  8586         $DS_ENDSUB
                                     1192       T12_S8_X:
     00010038 9F   00000144'EF   FA  1192                       CALLG   $S$, @#DS$ENDSUB
                                     119D  8587
                                     119D  8588         $DS_PAGE
    119D                  .SBTTL  SUBTEST 12.9:   WRITE RECORD - PADDED ZEROS
    119D  8591 ;+
    119D  8592 ; SUBTEST DESCRIPTION:
    119D  8593 ;
    119D  8594 ;          This subtest sends the TU58 a series of command packets to check
    119D  8595 ;          that a record written with less than 128 bytes of data, will
    119D  8596 ;          pad the remainder of the record with data bytes of zeroes. This
    119D  8597 ;          is accomplished by writing two consecutive records each with
    119D  8598 ;          128 bytes of all ones. Then a command packet is sent to
    119D  8599 ;          write the first record with one byte of alternating ones and
    119D  8600 ;          zeroes data. This command should write the first byte with
    119D  8601 ;          10101010 and the remaining 127 bytes with 00000000. The second
    119D  8602 ;          block should still contain 128 bytes of all ones data. This data
    119D  8603 ;          is read and checked. All end packets and data packets are checked as
    119D  8604 ;          the subtest progresses. All byte transactions between the host and TU58
    119D  8605 ;          are timed. If any errors are encountered, they are reported.
    119D  8606 ;
    119D  8607 ;
    119D  8608 ; SUBTEST STEPS:
    119D  8609 ;
    119D  8610 ;
    119D  8611 ;          SUBTEST WRITE RECORD PADDED ZEROS
    119D  8612 ;          :  Clear buffer
    119D  8613 ;          :  Point to first write command packet
    119D  8614 ;          :  DO for count = 1 to 3
    119D  8615 ;          :  :  REPEAT
    119D  8616 ;          :  :  :  Send command packet byte at a time
    119D  8617 ;          :  :  :  Set timer
    119D  8618 ;          :  :  :  IF timer expired AND no response
    119D  8619 ;          :  :  :  :  THEN
    119D  8620 ;          :  :  :  :  Report no response error
    119D  8621 ;          :  :  :  :  Exit test
    119D  8622 ;          :  :  :  ENDIF
    119D  8623 ;          :  :  : UNTIL all bytes sent
    119D  8624 ;          :  :  ENDREPEAT
    119D  8625 ;          :  :  Wait for continue
    119D  8626 ;          :  :  Set timer
    119D  8627 ;          :  :  IF timer expired AND no continue
    119D  8628 ;          :  :  :  THEN
    119D  8629 ;          :  :  :  Report no continue error
    119D  8630 ;          :  :  :  Exit test
    119D  8631 ;          :  :  ENDIF
    119D  8632 ;          :  :  Point to data packet
    119D  8633 ;          :  :  REPEAT
    119D  8634 ;          :  :  :  Send data packet byte at a time
    119D  8635 ;          :  :  :  Set timer
    119D  8636 ;          :  :  :  IF timer expired AND no response
    119D  8637 ;          :  :  :  :  THEN
    119D  8638 ;          :  :  :  :  Report no response error
    119D  8639 ;          :  :  :  :  Exit test
    119D  8640 ;          :  :  :  ENDIF
    119D  8641 ;          :  :  : UNTIL all bytes sent
    119D  8642 ;          :  :  ENDREPEAT
    119D  8643 ;          :  :  REPEAT
    119D  8644 ;          :  :  :  Set timer
```

```
                     119D  8645 ;        :  :  :  IF timer expired AND done set
                     119D  8646 ;        :  :  :  :  THEN
                     119D  8647 ;        :  :  :  :  :  Read byte
                     119D  8648 ;        :  :  :  ENDIF
                     119D  8649 ;        :  :  : UNTIL timer expired AND done clear
                     119D  8650 ;        :  :  ENDREPEAT
                     119D  8651 ;        :  :  IF expected byte count NEQ actual byte count
                     119D  8652 ;        :  :  :  THEN
                     119D  8653 ;        :  :  :  :  Report error
                     119D  8654 ;        :  :  :  :  Exit test
                     119D  8655 ;        :  :  ENDIF
                     119D  8656 ;        :  :  REPEAT
                     119D  8657 ;        :  :  :  Point to expected end packet
                     119D  8658 ;        :  :  :  IF expected end data NEQ received end data
                     119D  8659 ;        :  :  :  :  THEN
                     119D  8660 ;        :  :  :  :  Report end packet data error
                     119D  8661 ;        :  :  :  ENDIF
                     119D  8662 ;        :  :  : UNTIL all data checked
                     119D  8663 ;        :  :  ENDREPEAT
                     119D  8664 ;        :  :  Point to next write command
                     119D  8665 ;        :  ENDDO
                     119D  8666 ;        :  Clear buffer
                     119D  8667 ;        :  Point to first read command packet
                     119D  8668 ;        :  DO for count = 1 to 2
                     119D  8669 ;        :  :  REPEAT
                     119D  8670 ;        :  :  :  Send command packet byte at a time
                     119D  8671 ;        :  :  :  Set timer
                     119D  8672 ;        :  :  :  IF timer expired and no response
                     119D  8673 ;        :  :  :  :  THEN
                     119D  8674 ;        :  :  :  :  Report no response error
                     119D  8675 ;        :  :  :  :  Exit test
                     119D  8676 ;        :  :  :  ENDIF
                     119D  8677 ;        :  :  : UNTIL all bytes sent
                     119D  8678 ;        :  :  ENDREPEAT
                     119D  8679 ;        :  :  REPEAT
                     119D  8680 ;        :  :  :  Set timer
                     119D  8681 ;        :  :  :  IF timer expired AND done set
                     119D  8682 ;        :  :  :  :  THEN
                     119D  8683 ;        :  :  :  :  :  Read byte
                     119D  8684 ;        :  :  :  ENDIF
                     119D  8685 ;        :  :  : UNTIL timer expired AND done clear
                     119D  8686 ;        :  :  ENDREPEAT
                     119D  8687 ;        :  :  IF expected byte count NEQ actual byte count
                     119D  8688 ;        :  :  :  THEN
                     119D  8689 ;        :  :  :  :  Report error
                     119D  8690 ;        :  :  :  :  Exit test
                     119D  8691 ;        :  :  ENDIF
                     119D  8692 ;        :  :  Point to expected data packet
                     119D  8693 ;        :  :  REPEAT
                     119D  8694 ;        :  :  :  IF data expected NEQ data received
                     119D  8695 ;        :  :  :  :  THEN
                     119D  8696 ;        :  :  :  :  Report data error
                     119D  8697 ;        :  :  :  :  Exit subtest
                     119D  8698 ;        :  :  :  ENDIF
                     119D  8699 ;        :  :  :  Point to next byte of data
```

```
                               119D  8700 ;          :  :  : UNTIL all data in packet checked
                               119D  8701 ;          :  :  ENDREPEAT
                               119D  8702 ;          :  :  Point to expected end packet
                               119D  8703 ;          :  :  REPEAT
                               119D  8704 ;          :  :  :  IF end data expected NEQ end data received
                               119D  8705 ;          :  :  :  :  THEN
                               119D  8706 ;          :  :  :  :  Report end packet data error
                               119D  8707 ;          :  :  :  :  Exit subtest
                               119D  8708 ;          :  :  :  ENDIF
                               119D  8709 ;          :  :  : UNTIL all end packet bytes checked
                               119D  8710 ;          :  :  ENDREPEAT
                               119D  8711 ;          :  ENDDO
                               119D  8712 ;          ENDSUBTEST WRITE RECORD PADDED ZEROS
                               119D  8713 ;
                               119D  8714 ;
                               119D  8715 ;
                               119D  8716 ;.page
                               119D  8717 ;-
                               119D  8718           $DS_BGNSUB
                               119D       T12_S9::
 00010030 9F   00000150'EF  FA 119D                      CALLG   $S$, @#DS$BGNSUB
                               11A8  8719
               00000004'EF  94 11A8  8720           CLRB     B_COUNT1                ; initialize command counter
 0000000E'EF   00000084 8F  D0 11AE  8721           MOVL     #132,L_COUNT3           ; init byte count
          53   0000161F'EF  9E 11B9  8722           MOVAB    MCOMMAND_TBL_2,R3       ; load pointer to command table
                               11C0  8723                                            ; clear buffers
               00000000'EF  94 11C0  8724 10$:      CLRB     B_FLAGS                 ; initialize flags
          58   00000000'FF  9E 11C6  8725           MOVAB    @A_BUFFER_BEGIN,R8      ; load buffer starting address
          59   00000600 8F  D0 11CD  8726           MOVL     #BUFFER_SIZE,R9         ; do all pages
               00002134'EF  16 11D4  8727           JSB      CLEAR_BUFFER            ; go clear buffers
                               11DA  8728                                            ; send write command
          54   FFF3CB00 8F  D0 11DA  8729           MOVL     #DELAY_TIME3,R4         ; load send packet delay time
                    55   93 DE 11E1  8730           MOVAL    @(R3)+,R5               ; point to write command packet
                    57   0E D0 11E4  8731           MOVL     #14,R7                  ; load transfer byte count
               0000203D'EF  16 11E7  8732           JSB      SEND_TU58               ; send command packet
       24 00000000'EF    00 E1 11ED  8733           BBC      #V_CLOCK_FLAG,B_FLAGS,30$ ; br = no timeout error
                               11F5  8734           $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=NO_READY
               00001BEB'EF  DF 11F5                      PUSHAL  NO_READY
               0000190D'EF  DF 11FB                      PUSHAL  T_WR_REC_ERR
                         00 DD 1201                      PUSHL   #0
                         01 DD 1203                      PUSHL   #$ER
                               1205       ;;; TEST 12, SUBTEST 9, ERROR 1
          000100D0 9F    04 FB 1205                      CALLS   #$$M, @#DS$ERRHARD
                               120C  8735           $DS_BNCOMPLETE  20$
                      03 50 E9 120C                      BLBC    R0, 20$
                               120F  8736           $DS_ABORT       TEST
                         50 D4 120F                      CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                            04 1211                      RET                     ; TERMINATE TEST
                               1212  8737 20$:      $DS_ABORT
          00010020 9F    6C FA 1212                      CALLG   (AP), @#DS$ABORT
                               1219  8738                                            ; read continue
          54   FFF3CB00 8F  D0 1219  8739 30$:      MOVL     #DELAY_TIME3,R4         ; load receive packet delay time
          55   00000000'FF  9E 1220  8740           MOVAB    @A_BUFFER_BEGIN,R5      ; load buffer address to store data
             57   55     01 C1 1227  8741           ADDL3    #1,R5,R7                ; load last expected address of data
               000020C8'EF  16 122B  8742           JSB      MRECEIVE_TU58           ; go get data from TU58
```

```
         5A   00000023'EF   9E  1231  8743          MOVAB   B_CONTINUE,R10          ; load address of expected data
         5B   00000000'FF   9E  1238  8744          MOVAB   @A_BUFFER_BEGIN,R11     ; load address of actual data
                   57  55   D1  123F  8745          CMPL    R5,R7                   ; check for correct number of bytes
                       2A   13  1242  8746          BEQL    70$                     ; br = correct number of bytes
                                1244  8747 35$:     $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=REC_ERR,-
                                1244  8748                           P1=#T_REC_CONT
              00001D47'8F   DD  1244                        PUSHL   #T_REC_CONT
              00001EB4'EF   DF  124A                        PUSHAL  REC_ERR
              0000190D'EF   DF  1250                        PUSHAL  T_WR_REC_ERR
                       00   DD  1256                        PUSHL   #0
                       02   DD  1258                        PUSHL   #$ER
                                125A        ;;; TEST 12, SUBTEST 9, ERROR 2
         000100D0 9F   05   FB  125A                        CALLS   #$$M, @#DS$ERRHARD
                                1261  8749          $DS_BNCOMPLETE  40$
                    03 50   E9  1261                        BLBC    R0, 40$
                                1264  8750          $DS_ABORT       TEST
                       50   D4  1264                        CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                       04       1266                        RET                     ; TERMINATE TEST
                                1267  8751
                                1267  8752 40$:     $DS_ABORT
         00010020 9F   6C   FA  1267                        CALLG   (AP), @#DS$ABORT
                                126E  8753                                          ; check continue
              5C       01   D0  126E  8754 70$:     MOVL    #1,R12                  ; load number of bytes to check
                                1271  8755          $DS_BREAK
         00010058 9F   6E   FA  1271                        CALLG   (SP), @#DS$BREAK
              0000213B'EF   16  1278  8756          JSB     CHECK_PACKET            ; go check continue
                                127E  8757          $DS_BNCOMPLETE  90$             ; br = no error
                    24 50   E9  127E                        BLBC    R0, 90$
                                1281  8758          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=DATA_ERR_CONT
              00001E4D'EF   DF  1281                        PUSHAL  DATA_ERR_CONT
              0000190D'EF   DF  1287                        PUSHAL  T_WR_REC_ERR
                       00   DD  128D                        PUSHL   #0
                       03   DD  128F                        PUSHL   #$ER
                                1291        ;;; TEST 12, SUBTEST 9, ERROR 3
         000100D0 9F   04   FB  1291                        CALLS   #$$M, @#DS$ERRHARD
                                1298  8759          $DS_BNCOMPLETE  80$
                    03 50   E9  1298                        BLBC    R0, 80$
                                129B  8760          $DS_ABORT       TEST
                       50   D4  129B                        CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                       04       129D                        RET                     ; TERMINATE TEST
                                129E  8761 80$:     $DS_ABORT
         00010020 9F   6C   FA  129E                        CALLG   (AP), @#DS$ABORT
                                12A5  8762                                          ; send data packet
         54   FFFB6C20 8F   D0  12A5  8763 90$:     MOVL    #DELAY_TIME1,R4         ; load send packet delay time
                   55  93   DE  12AC  8764          MOVAL   @(R3)+,R5               ; point to data packet
         57   0000000E'EF   D0  12AF  8765          MOVL    L_COUNT3,R7             ; load transfer byte count
              0000203D'EF   16  12B6  8766          JSB     SEND_TU58               ; send data packet
      24 00000000'EF   00   E1  12BC  8767          BBC     #V_CLOCK_FLAG,B_FLAGS,110$ ; br = no timeout error
                                12C4  8768          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=NO_READY
              00001BEB'EF   DF  12C4                        PUSHAL  NO_READY
              0000190D'EF   DF  12CA                        PUSHAL  T_WR_REC_ERR
                       00   DD  12D0                        PUSHL   #0
                       04   DD  12D2                        PUSHL   #$ER
                                12D4        ;;; TEST 12, SUBTEST 9, ERROR 4
         000100D0 9F   04   FB  12D4                        CALLS   #$$M, @#DS$ERRHARD
```

```
                                     12DB  8769         $DS_BNCOMPLETE  100$
                         03 50   E9  12DB                       BLBC    R0, 100$
                                     12DE  8770         $DS_ABORT       TEST
                            50   D4  12DE                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  12E0                       RET                     ; TERMINATE TEST
                                     12E1  8771 100$:   $DS_ABORT
              00010020 9F   6C   FA  12E1                       CALLG   (AP), @#DS$ABORT
                                     12E8  8772
                   00000000'FF   94  12E8  8773 110$:   CLRB    @A_BUFFER_BEGIN          ; clear buffer used to store continue
                                     12EE  8774                                          ; read end packet
              54   FFC2F700 8F   D0  12EE  8775         MOVL    #DELAY_TIME6,R4          ; load receive packet delay time
              55   00000000'FF   9E  12F5  8776         MOVAB   @A_BUFFER_BEGIN,R5       ; load buffer address to store data
                  57   55   0E   C1  12FC  8777         ADDL3   #14,R5,R7                ; load last expected address of data
                   000020C8'EF   16  1300  8778         JSB     MRECEIVE_TU58            ; go get data from TU58
                       5A   93   DE  1306  8779         MOVAL   @(R3)+,R10               ; load address of expected end packet
              5B   00000000'FF   9E  1309  8780         MOVAB   @A_BUFFER_BEGIN,R11      ; load address of actual end packet
                       57   55   D1  1310  8781         CMPL    R5,R7                    ; check for correct number of bytes
                            31   13  1313  8782         BEQL    150$                     ; br = correct number of bytes
              00000000'EF   08   88  1315  8783         BISB2   #M_END_FLAG,B_FLAGS      ; set end error flag
                                     131C  8784 115$:   $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=REC_ERR,-
                                     131C  8785                         P1=#T_REC_END
                   00001D2E'8F   DD  131C                       PUSHL   #T_REC_END
                   00001EB4'EF   DF  1322                       PUSHAL  REC_ERR
                   0000190D'EF   DF  1328                       PUSHAL  T_WR_REC_ERR
                            00   DD  132E                       PUSHL   #0
                            05   DD  1330                       PUSHL   #$ER
                                     1332       ;;; TEST 12, SUBTEST 9, ERROR 5
              000100D0 9F   05   FB  1332                       CALLS   #$$M, @#DS$ERRHARD
                                     1339  8786         $DS_BNCOMPLETE  120$
                         03 50   E9  1339                       BLBC    R0, 120$
                                     133C  8787         $DS_ABORT       TEST
                            50   D4  133C                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  133E                       RET                     ; TERMINATE TEST
                                     133F  8788
                                     133F  8789 120$:   $DS_ABORT
              00010020 9F   6C   FA  133F                       CALLG   (AP), @#DS$ABORT
                                     1346  8790                                          ; check end packet
                       5C   0E   D0  1346  8791 150$:   MOVL    #14,R12                  ; load number of bytes to check
                                     1349  8792         $DS_BREAK
              00010058 9F   6E   FA  1349                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16  1350  8793         JSB     CHECK_PACKET             ; go check end packet
                                     1356  8794         $DS_BNCOMPLETE  160$             ; br = no error
                         24 50   E9  1356                       BLBC    R0, 160$
                                     1359  8795         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=DATA_ERR_END
                   00001E9D'EF   DF  1359                       PUSHAL  DATA_ERR_END
                   0000190D'EF   DF  135F                       PUSHAL  T_WR_REC_ERR
                            00   DD  1365                       PUSHL   #0
                            06   DD  1367                       PUSHL   #$ER
                                     1369       ;;; TEST 12, SUBTEST 9, ERROR 6
              000100D0 9F   04   FB  1369                       CALLS   #$$M, @#DS$ERRHARD
                                     1370  8796         $DS_BNCOMPLETE  155$
                         03 50   E9  1370                       BLBC    R0, 155$
                                     1373  8797         $DS_ABORT       TEST
                            50   D4  1373                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  1375                       RET                     ; TERMINATE TEST
```

```
                                1376  8798 155$:    $DS_ABORT
      00010020 9F    6C    FA   1376                        CALLG   (AP), a#DS$ABORT
      14 0000000E'EF    E8   137D  8799 160$:    BLBS    L_COUNT3,165$           ; go do reads if writes done
 FE32 00000004'EF   01    01    9D   1384  8800             ACBB    #1,#1,B_COUNT1,10$      ; br = not finished
      0000000E'EF    05    D0   138E  8801             MOVL    #5,L_COUNT3             ; load byte count for last write
                 FE28    31   1395  8802             BRW     10$                     ; go do last write
          00000004'EF    94   1398  8803 165$:    CLRB    B_COUNT1                ; initialize command counter
                              139E  8804                                             ; clear buffers
          00000000'EF    94   139E  8805 170$:    CLRB    B_FLAGS                 ; initialize flags
      58  00000000'FF    9E   13A4  8806             MOVAB   aA_BUFFER_BEGIN,R8      ; load buffer starting address
      59  00000600 8F    D0   13AB  8807             MOVL    #BUFFER_SIZE,R9         ; do all pages
          00002134'EF    16   13B2  8808             JSB     CLEAR_BUFFER            ; go clear buffers
                              13B8  8809                                             ; send read command
      54  FFFB6C20 8F    D0   13B8  8810             MOVL    #DELAY_TIME1,R4         ; load send packet delay time
              55   93    DE   13BF  8811             MOVAL   a(R3)+,R5               ; point to read command packet
              57   0E    D0   13C2  8812             MOVL    #14,R7                  ; load transfer byte count
          0000203D'EF    16   13C5  8813             JSB     SEND_TU58               ; send command packet
   24 00000000'EF    00    E1   13CB  8814             BBC     #V_CLOCK_FLAG,B_FLAGS,190$ ; br = no timeout error
                              13D3  8815             $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=NO_READY
          00001BEB'EF    DF   13D3                        PUSHAL  NO_READY
          00001931'EF    DF   13D9                        PUSHAL  T_RD_REC_ERR
                   00    DD   13DF                        PUSHL   #0
                   07    DD   13E1                        PUSHL   #$ER
                              13E3       ;;; TEST 12, SUBTEST 9, ERROR 7
      000100D0 9F    04    FB   13E3                        CALLS   #$$M, a#DS$ERRHARD
                              13EA  8816             $DS_BNCOMPLETE  180$
              03 50      E9   13EA                        BLBC    R0, 180$
                              13ED  8817             $DS_ABORT       TEST
                   50    D4   13ED                        CLRL    R0                ; SEND A WARNING TO THE SUPERVISOR
                   04         13EF                        RET                       ; TERMINATE TEST
                              13F0  8818 180$:    $DS_ABORT
      00010020 9F    6C    FA   13F0                        CALLG   (AP), a#DS$ABORT
                              13F7  8819                                             ; read data and end packets
      54  FFF3CB00 8F    D0   13F7  8820 190$:    MOVL    #DELAY_TIME3,R4         ; load receive packet delay time
      55  00000000'FF    9E   13FE  8821             MOVAB   aA_BUFFER_BEGIN,R5      ; load buffer address to store data
  57  55  00000092 8F    C1   1405  8822             ADDL3   #146,R5,R7              ; load last expected address of data
          000020C8'EF    16   140D  8823             JSB     MRECEIVE_TU58           ; go get data from TU58
      5B  00000000'FF    9E   1413  8824             MOVAB   aA_BUFFER_BEGIN,R11     ; load address of actual data
              5A   93    DE   141A  8825             MOVAL   a(R3)+,R10              ; load address of expected data
              57   55    D1   141D  8826             CMPL    R5,R7                   ; check for correct number of bytes
              31   13         1420  8827             BEQL    230$                    ; br = correct number of bytes
      00000000'EF    04    88   1422  8828             BISB2   #M_DATA_FLAG,B_FLAGS    ; set data error flag
                              1429  8829 195$:    $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=REC_ERR,-
                              1429  8830                               P1=#T_REC_DATA
          00001D68'8F    DD   1429                        PUSHL   #T_REC_DATA
          00001EB4'EF    DF   142F                        PUSHAL  REC_ERR
          00001931'EF    DF   1435                        PUSHAL  T_RD_REC_ERR
                   00    DD   143B                        PUSHL   #0
                   08    DD   143D                        PUSHL   #$ER
                              143F       ;;; TEST 12, SUBTEST 9, ERROR 8
      000100D0 9F    05    FB   143F                        CALLS   #$$M, a#DS$ERRHARD
                              1446  8831             $DS_BNCOMPLETE  200$
              03 50      E9   1446                        BLBC    R0, 200$
                              1449  8832             $DS_ABORT       TEST
                   50    D4   1449                        CLRL    R0                ; SEND A WARNING TO THE SUPERVISOR
```

```
                                 04  144B                       RET                      ; TERMINATE TEST
                                     144C  8833
                                     144C  8834 200$:   $DS_ABORT
              00010020 9F   6C   FA  144C                       CALLG   (AP), @#DS$ABORT
                                     1453  8835                                          ; check data packet
              5C   00000084 8F   D0  1453  8836 230$:   MOVL    #132,R12                 ; load number of bytes to check
                                     145A  8837 240$:   $DS_BREAK
              00010058 9F   6E   FA  145A                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16  1461  8838         JSB     CHECK_PACKET             ; go check data packet
                                     1467  8839         $DS_BNCOMPLETE  260$             ; br = no error
                         24 50   E9  1467                       BLBC    R0, 260$
                                     146A  8840         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=DATA_ERR_DATA
                   00001E75'EF   DF  146A                       PUSHAL  DATA_ERR_DATA
                   00001931'EF   DF  1470                       PUSHAL  T_RD_REC_ERR
                            00   DD  1476                       PUSHL   #0
                            09   DD  1478                       PUSHL   #$ER
                                     147A       ;;; TEST 12, SUBTEST 9, ERROR 9
              000100D0 9F   04   FB  147A                       CALLS   #$$M, @#DS$ERRHARD
                                     1481  8841         $DS_BNCOMPLETE  250$
                         03 50   E9  1481                       BLBC    R0, 250$
                                     1484  8842         $DS_ABORT       TEST
                            50   D4  1484                       CLRL    R0               ; SEND A WARNING TO THE SUPERVISOR
                                 04  1486                       RET                      ; TERMINATE TEST
                                     1487  8843 250$:   $DS_ABORT
              00010020 9F   6C   FA  1487                       CALLG   (AP), @#DS$ABORT
                                     148E  8844                                          ; check end packet
              5B   00000084 8F   C0  148E  8845 260$:   ADDL2   #132,R11                 ; adjust pointer to next packet
                       5A   93   DE  1495  8846         MOVAL   @(R3)+,R10               ; load address of expected end packet
                       5C   0E   D0  1498  8847         MOVL    #14,R12                  ; load number of bytes to check
                                     149B  8848         $DS_BREAK
              00010058 9F   6E   FA  149B                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF   16  14A2  8849         JSB     CHECK_PACKET             ; go check end packet
                                     14A8  8850         $DS_BNCOMPLETE  270$             ; br = no error
                         24 50   E9  14A8                       BLBC    R0, 270$
                                     14AB  8851         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=DATA_ERR_END
                   00001E9D'EF   DF  14AB                       PUSHAL  DATA_ERR_END
                   00001931'EF   DF  14B1                       PUSHAL  T_RD_REC_ERR
                            00   DD  14B7                       PUSHL   #0
                            0A   DD  14B9                       PUSHL   #$ER
                                     14BB       ;;; TEST 12, SUBTEST 9, ERROR 10
              000100D0 9F   04   FB  14BB                       CALLS   #$$M, @#DS$ERRHARD
                                     14C2  8852         $DS_BNCOMPLETE  265$
                         03 50   E9  14C2                       BLBC    R0, 265$
                                     14C5  8853         $DS_ABORT       TEST
                            50   D4  14C5                       CLRL    R0               ; SEND A WARNING TO THE SUPERVISOR
                                 04  14C7                       RET                      ; TERMINATE TEST
                                     14C8  8854 265$:   $DS_ABORT
              00010020 9F   6C   FA  14C8                       CALLG   (AP), @#DS$ABORT
    FEC5 00000004'EF   01   01   9D  14CF  8855 270$:   ACBB    #1,#1,B_COUNT1,170$      ; br = not finished reading
                                     14D9  8856
                                     14D9  8857
                                     14D9  8858
                                     14D9  8859
                                     14D9  8860         $DS_ENDSUB
                                     14D9       T12_S9_X:
```

```
    00010038 9F   00000150'EF   FA  14D9                    CALLG   $S$, a#DS$ENDSUB
                                    14E4  8861
                                    14E4  8862          $DS_PAGE
```

```
14E4            .SBTTL  SUBTEST 12.10:   BLOCK WRITE - READ
14E4    ;+
14E4    ; SUBTEST DESCRIPTION:
14E4    ;
14E4    ;       This subtest sends the TU58 controller a series of command packets
14E4    ;       that will cause head switches, forward seeks, reverse seeks and
14E4    ;       seeks with head switches during read and write operations on
14E4    ;       blocks. All data written is read and checked. All end packets
14E4    ;       are checked as the subtest progresses. If any errors occur they are
14E4    ;       reported.
14E4    ;
14E4    ;
14E4    ; SUBTEST STEPS:
14E4    ;
14E4    ;       SUBTEST BLOCK WRITE - READ
14E4    ;       :  DO for count = 1 to 5
14E4    ;       :  :  Point to write command packet
14E4    ;       :  :  Clear buffer
14E4    ;       :  :  REPEAT
14E4    ;       :  :  :  Send command packet byte at a time
14E4    ;       :  :  :  Set timer
14E4    ;       :  :  :  IF timer expired AND no response
14E4    ;       :  :  :  :  THEN
14E4    ;       :  :  :  :  Report no response error
14E4    ;       :  :  :  :  Exit test
14E4    ;       :  :  :  ENDIF
14E4    ;       :  :  : UNTIL all bytes sent
14E4    ;       :  :  ENDREPEAT
14E4    ;       :  :  DO for count = 1 to 4
14E4    ;       :  :  :  Wait for continue
14E4    ;       :  :  :  Set timer
14E4    ;       :  :  :  IF timer expired AND no continue
14E4    ;       :  :  :  :  THEN
14E4    ;       :  :  :  :  Report no continue error
14E4    ;       :  :  :  :  Exit test
14E4    ;       :  :  :  ENDIF
14E4    ;       :  :  :  Point to data packet
14E4    ;       :  :  :  REPEAT
14E4    ;       :  :  :  :  Send data packet byte at a time
14E4    ;       :  :  :  :  Set timer
14E4    ;       :  :  :  :  IF timer expired AND no response
14E4    ;       :  :  :  :  :  THEN
14E4    ;       :  :  :  :  :  Report no response error
14E4    ;       :  :  :  :  :  Exit test
14E4    ;       :  :  :  :  ENDIF
14E4    ;       :  :  :  : UNTIL all bytes sent
14E4    ;       :  :  :  ENDREPEAT
14E4    ;       :  :  ENDDO
14E4    ;       :  :  REPEAT
14E4    ;       :  :  :  Set timer
14E4    ;       :  :  :  IF timer expired AND done set
14E4    ;       :  :  :  :  THEN
14E4    ;       :  :  :  :  :  Read byte
14E4    ;       :  :  :  ENDIF
14E4    ;       :  :  : UNTIL timer expired AND done clear
```

```
14E4  8919 ;       :  :  ENDREPEAT
14E4  8920 ;       :  :  IF expected byte count NEQ actual byte count
14E4  8921 ;       :  :  :  THEN
14E4  8922 ;       :  :  :  :  Report error
14E4  8923 ;       :  :  :  :  Exit test
14E4  8924 ;       :  :  ENDIF
14E4  8925 ;       :  :  REPEAT
14E4  8926 ;       :  :  :  Point to expected end packet
14E4  8927 ;       :  :  :  IF expected end data NEQ received end data
14E4  8928 ;       :  :  :  :  THEN
14E4  8929 ;       :  :  :  :  Report end packet data error
14E4  8930 ;       :  :  :  ENDIF
14E4  8931 ;       :  :  : UNTIL all data checked
14E4  8932 ;       :  :  ENDREPEAT
14E4  8933 ;       :  :  Point to read command packet
14E4  8934 ;       :  :  REPEAT
14E4  8935 ;       :  :  :  Send command packet byte at a time
14E4  8936 ;       :  :  :  Set timer
14E4  8937 ;       :  :  :  IF timer expired and no response
14E4  8938 ;       :  :  :  :  THEN
14E4  8939 ;       :  :  :  :  Report no response error
14E4  8940 ;       :  :  :  :  Exit test
14E4  8941 ;       :  :  :  ENDIF
14E4  8942 ;       :  :  : UNTIL all bytes sent
14E4  8943 ;       :  :  ENDREPEAT
14E4  8944 ;       :  :  REPEAT
14E4  8945 ;       :  :  :  Set timer
14E4  8946 ;       :  :  :  IF timer expired AND done set
14E4  8947 ;       :  :  :  :  THEN
14E4  8948 ;       :  :  :  :  :  Read byte
14E4  8949 ;       :  :  :  ENDIF
14E4  8950 ;       :  :  : UNTIL timer expired AND done clear
14E4  8951 ;       :  :  ENDREPEAT
14E4  8952 ;       :  :  IF expected byte count NEQ actual byte count
14E4  8953 ;       :  :  :  THEN
14E4  8954 ;       :  :  :  :  Report error
14E4  8955 ;       :  :  :  :  Exit test
14E4  8956 ;       :  :  ENDIF
14E4  8957 ;       :  :  REPEAT
14E4  8958 ;       :  :  :  Point to expected data packet
14E4  8959 ;       :  :  :  REPEAT
14E4  8960 ;       :  :  :  :  IF data expected NEQ data received
14E4  8961 ;       :  :  :  :  :  THEN
14E4  8962 ;       :  :  :  :  :  Report data error
14E4  8963 ;       :  :  :  :  :  Exit subtest
14E4  8964 ;       :  :  :  :  ENDIF
14E4  8965 ;       :  :  :  :  Point to next byte of data
14E4  8966 ;       :  :  :  : UNTIL all data in packet checked
14E4  8967 ;       :  :  :  ENDREPEAT
14E4  8968 ;       :  :  : UNTIL 4 data packets checked
14E4  8969 ;       :  :  ENDREPEAT
14E4  8970 ;       :  :  Point to expected end packet
14E4  8971 ;       :  :  REPEAT
14E4  8972 ;       :  :  :  IF data expected NEQ data received
14E4  8973 ;       :  :  :  :  THEN
```

```
                                          14E4   8974 ;          :  :  :  :  Report data error
                                          14E4   8975 ;          :  :  :  :  Exit subtest
                                          14E4   8976 ;          :  :  :  ENDIF
                                          14E4   8977 ;          :  :  : UNTIL all end packet bytes checked
                                          14E4   8978 ;          :  :  ENDREPEAT
                                          14E4   8979 ;          :  ENDDO
                                          14E4   8980 ;          ENDSUBTEST BLOCK WRITE - READ
                                          14E4   8981 ;
                                          14E4   8982 ;
                                          14E4   8983 ;.page
                                          14E4   8984 ;-
                                          14E4   8985            $DS_BGNSUB
                                          14E4        T12_S10::
      00010030 9F    0000015C'EF    FA    14E4                           CALLG    $S$, @#DS$BGNSUB
                     00000001'EF    94    14EF   8986            CLRB     B_FLAGS2                  ; initialize repeat flag
             00000001'EF    06      88    14F5   8987            BISB2    #<M_REPEAT_SEND+M_REPEAT_CHECK>,B_FLAGS2
                     00000004'EF    94    14FC   8988            CLRB     B_COUNT1                  ; initialize command counter
             53      0000165B'EF    9E    1502   8989            MOVAB    MCOMMAND_TBL_3,R3         ; load pointer to command table
                                          1509   8990                                               ; clear buffers
                     00000000'EF    94    1509   8991 10$:       CLRB     B_FLAGS                   ; initialize flags
             58      00000000'FF    9E    150F   8992            MOVAB    @A_BUFFER_BEGIN,R8        ; load buffer starting address
             59      00000600 8F    D0    1516   8993            MOVL     #BUFFER_SIZE,R9           ; do all pages
                     00002134'EF    16    151D   8994            JSB      CLEAR_BUFFER              ; go clear buffers
                                          1523   8995                                               ; send write command
             54      FFF3CB00 8F    D0    1523   8996            MOVL     #DELAY_TIME3,R4           ; load send packet delay time
                           55   93  DE    152A   8997            MOVAL    @(R3)+,R5                 ; point to write command packet
                           57   0E  D0    152D   8998            MOVL     #14,R7                    ; load transfer byte count
                     0000203D'EF    16    1530   8999            JSB      SEND_TU58                 ; send command packet
          24 00000000'EF    00      E1    1536   9000            BBC      #V_CLOCK_FLAG,B_FLAGS,30$ ; br = no timeout error
                                          153E   9001            $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=NO_READY
                     00001BEB'EF    DF    153E                           PUSHAL   NO_READY
                     000018C8'EF    DF    1544                           PUSHAL   T_WR_BLK_ERR
                              00    DD    154A                           PUSHL    #0
                              01    DD    154C                           PUSHL    #$ER
                                          154E        ;;; TEST 12, SUBTEST 10, ERROR 1
             000100D0 9F    04      FB    154E                           CALLS    #$$M, @#DS$ERRHARD
                                          1555   9002            $DS_BNCOMPLETE  20$
                           03 50    E9    1555                           BLBC     R0, 20$
                                          1558   9003            $DS_ABORT        TEST
                              50    D4    1558                           CLRL     R0                ; SEND A WARNING TO THE SUPERVISOR
                              04          155A                           RET                        ; TERMINATE TEST
                                          155B   9004 20$:       $DS_ABORT
             00010020 9F    6C      FA    155B                           CALLG    (AP), @#DS$ABORT
                                          1562   9005                                               ; read continue
                     00000003'EF    94    1562   9006 30$:       CLRB     B_COUNT                   ; initialize data packet counter
             54      FF7E4CE0 8F    D0    1568   9007 35$:       MOVL     #DELAY_TIME7,R4           ; load receive packet delay time
             55      00000000'FF    9E    156F   9008            MOVAB    @A_BUFFER_BEGIN,R5        ; load buffer address to store data
                 57   55    01      C1    1576   9009            ADDL3    #1,R5,R7                  ; load last expected address of data
                     000020C8'EF    16    157A   9010            JSB      MRECEIVE_TU58             ; go get data from TU58
             5A      00000023'EF    9E    1580   9011            MOVAB    B_CONTINUE,R10            ; load address of expected continue
             5B      00000000'FF    9E    1587   9012            MOVAB    @A_BUFFER_BEGIN,R11       ; load address of buffer
                           57   55  D1    158E   9013            CMPL     R5,R7                     ; check for correct number of bytes
                                2A  13    1591   9014            BEQL     70$                       ; br = correct number of bytes
                                          1593   9015 37$:       $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=REC_ERR,-
                                          1593   9016                             P1=#T_REC_CONT
          00001D47'8F   DD  1593                       PUSHL   #T_REC_CONT
          00001EB4'EF   DF  1599                       PUSHAL  REC_ERR
          000018C8'EF   DF  159F                       PUSHAL  T_WR_BLK_ERR
                   00   DD  15A5                       PUSHL   #0
                   02   DD  15A7                       PUSHL   #$ER
                            15A9           ;;; TEST 12, SUBTEST 10, ERROR 2
     000100D0 9F   05   FB  15A9                       CALLS   #$$M, @#DS$ERRHARD
                            15B0  9017          $DS_BNCOMPLETE  40$
                03 50   E9  15B0                       BLBC    R0, 40$
                            15B3  9018          $DS_ABORT       TEST
                   50   D4  15B3                       CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                   04       15B5                       RET                         ; TERMINATE TEST
                            15B6  9019
                            15B6  9020 40$:     $DS_ABORT
     00010020 9F   6C   FA  15B6                       CALLG   (AP), @#DS$ABORT
                            15BD  9021                                             ; check continue
              5C   01   D0  15BD  9022 70$:     MOVL    #1,R12                     ; load number of bytes to check
                            15C0  9023          $DS_BREAK
     00010058 9F   6E   FA  15C0                       CALLG   (SP), @#DS$BREAK
          0000213B'EF   16  15C7  9024          JSB     CHECK_PACKET               ; go check continue
                            15CD  9025          $DS_BNCOMPLETE  90$                ; br = no error
                24 50   E9  15CD                       BLBC    R0, 90$
                            15D0  9026          $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=DATA_ERR_CONT
          00001E4D'EF   DF  15D0                       PUSHAL  DATA_ERR_CONT
          000018C8'EF   DF  15D6                       PUSHAL  T_WR_BLK_ERR
                   00   DD  15DC                       PUSHL   #0
                   03   DD  15DE                       PUSHL   #$ER
                            15E0           ;;; TEST 12, SUBTEST 10, ERROR 3
     000100D0 9F   04   FB  15E0                       CALLS   #$$M, @#DS$ERRHARD
                            15E7  9027          $DS_BNCOMPLETE  80$
                03 50   E9  15E7                       BLBC    R0, 80$
                            15EA  9028          $DS_ABORT       TEST
                   50   D4  15EA                       CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                   04       15EC                       RET                         ; TERMINATE TEST
                            15ED  9029 80$:     $DS_ABORT
     00010020 9F   6C   FA  15ED                       CALLG   (AP), @#DS$ABORT
                            15F4  9030                                             ; send data packet
     54   FFFB6C20 8F   D0  15F4  9031 90$:     MOVL    #DELAY_TIME1,R4            ; load send packet delay time
               55  93   DE  15FB  9032          MOVAL   @(R3)+,R5                  ; point to data packet
     57   00000084 8F   D0  15FE  9033          MOVL    #132,R7                    ; load transfer byte count
          0000203D'EF   16  1605  9034          JSB     SEND_TU58                  ; send data packet
  24 00000000'EF   00   E1  160B  9035          BBC     #V_CLOCK_FLAG,B_FLAGS,110$ ; br = no timeout error
                            1613  9036          $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=NO_READY
          00001BEB'EF   DF  1613                       PUSHAL  NO_READY
          000018C8'EF   DF  1619                       PUSHAL  T_WR_BLK_ERR
                   00   DD  161F                       PUSHL   #0
                   04   DD  1621                       PUSHL   #$ER
                            1623           ;;; TEST 12, SUBTEST 10, ERROR 4
     000100D0 9F   04   FB  1623                       CALLS   #$$M, @#DS$ERRHARD
                            162A  9037          $DS_BNCOMPLETE  100$
                03 50   E9  162A                       BLBC    R0, 100$
                            162D  9038          $DS_ABORT       TEST
                   50   D4  162D                       CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                   04       162F                       RET                         ; TERMINATE TEST
                            1630  9039 100$:    $DS_ABORT
```

```
         00010020 9F    6C   FA  1630                           CALLG   (AP), @#DS$ABORT
                                 1637   9040
                00000000'FF   94  1637   9041 110$:   CLRB     @A_BUFFER_BEGIN          ; clear buffer used to store continue
    FF21 00000003'EF    01  03   9D  163D   9042              ACBB     #3,#1,B_COUNT,35$        ; br = not finished
          10 00000001'EF    01   E1  1647   9043 112$:   BBC      #V_REPEAT_SEND,B_FLAGS2,115$ ; br =  not repeating
                00000003'EF   94  164F   9044              CLRB     B_COUNT                  ; initialize counter to repeat
         00000001'EF    02   8A  1655   9045              BICB2    #M_REPEAT_SEND,B_FLAGS2  ; init repeat flag to stop repeating
                       FF09   31  165C   9046              BRW      35$                      ; finish sending data
                                 165F   9047                                                ; read end packet
         54    FFE17B80 8F   D0  165F   9048 115$:   MOVL     #DELAY_TIMES,R4          ; load receive packet delay time
         55    00000000'FF   9E  1666   9049              MOVAB    @A_BUFFER_BEGIN,R5       ; load buffer address to store data
            57    55    0E   C1  166D   9050              ADDL3    #14,R5,R7                ; load last expected address of data
                000020C8'EF   16  1671   9051              JSB      MRECEIVE_TU58            ; go get data from TU58
                      5A    93   DE  1677   9052              MOVAL    @(R3)+,R10               ; load address of expected end packet
         5B    00000000'FF   9E  167A   9053              MOVAB    @A_BUFFER_BEGIN,R11      ; load address of actual end packet
                      57    55   D1  1681   9054              CMPL     R5,R7                    ; check for correct number of bytes
                      31    13  1684   9055              BEQL     150$                     ; br = correct number of bytes
         00000000'EF    08   88  1686   9056              BISB2    #M_END_FLAG,B_FLAGS      ; set end error flag
                                 168D   9057 117$:   $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=REC_ERR,-
                                 168D   9058                               P1=#T_REC_END
                00001D2E'8F   DD  168D                          PUSHL    #T_REC_END
                00001EB4'EF   DF  1693                          PUSHAL   REC_ERR
                000018C8'EF   DF  1699                          PUSHAL   T_WR_BLK_ERR
                         00   DD  169F                          PUSHL    #0
                         05   DD  16A1                          PUSHL    #$ER
                                 16A3          ;;; TEST 12, SUBTEST 10, ERROR 5
         000100D0 9F    05   FB  16A3                          CALLS    #$$M, @#DS$ERRHARD
                                 16AA   9059              $DS_BNCOMPLETE   120$
                      03 50   E9  16AA                          BLBC     R0, 120$
                                 16AD   9060              $DS_ABORT        TEST
                         50   D4  16AD                          CLRL     R0                  ; SEND A WARNING TO THE SUPERVISOR
                         04  16AF                          RET                          ; TERMINATE TEST
                                 16B0   9061
                                 16B0   9062 120$:   $DS_ABORT
         00010020 9F    6C   FA  16B0                          CALLG    (AP), @#DS$ABORT
                                 16B7   9063                                                ; check end packet
            5C    0E   D0  16B7   9064 150$:   MOVL     #14,R12                  ; load number of bytes to check
                                 16BA   9065              $DS_BREAK
         00010058 9F    6E   FA  16BA                          CALLG    (SP), @#DS$BREAK
                0000213B'EF   16  16C1   9066              JSB      CHECK_PACKET             ; go check end packet
                                 16C7   9067              $DS_BNCOMPLETE   160$                ; br = no error
                      24 50   E9  16C7                          BLBC     R0, 160$
                                 16CA   9068              $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_BLK_ERR,PRLINK=DATA_ERR_END
                00001E9D'EF   DF  16CA                          PUSHAL   DATA_ERR_END
                000018C8'EF   DF  16D0                          PUSHAL   T_WR_BLK_ERR
                         00   DD  16D6                          PUSHL    #0
                         06   DD  16D8                          PUSHL    #$ER
                                 16DA          ;;; TEST 12, SUBTEST 10, ERROR 6
         000100D0 9F    04   FB  16DA                          CALLS    #$$M, @#DS$ERRHARD
                                 16E1   9069              $DS_BNCOMPLETE   155$
                      03 50   E9  16E1                          BLBC     R0, 155$
                                 16E4   9070              $DS_ABORT        TEST
                         50   D4  16E4                          CLRL     R0                  ; SEND A WARNING TO THE SUPERVISOR
                         04  16E6                          RET                          ; TERMINATE TEST
                                 16E7   9071 155$:   $DS_ABORT
         00010020 9F    6C    FA  16E7                       CALLG   (AP), @#DS$ABORT
                                  16EE  9072                                                   ; clear buffers
             00000000'EF    94    16EE  9073 160$:   CLRB    B_FLAGS                           ; initialize flags
         58  00000000'FF    9E    16F4  9074         MOVAB   @A_BUFFER_BEGIN,R8                ; load buffer starting address
         59  00000600 8F    D0    16FB  9075         MOVL    #BUFFER_SIZE,R9                   ; do all pages
             00002134'EF    16    1702  9076         JSB     CLEAR_BUFFER                      ; go clear buffers
                                  1708  9077                                                   ; send read command
         54  FFFB6C20 8F    D0    1708  9078         MOVL    #DELAY_TIME1,R4                   ; load send packet delay time
                   55 93    DE    170F  9079         MOVAL   @(R3)+,R5                         ; point to read command packet
                   57 0E    D0    1712  9080         MOVL    #14,R7                            ; load transfer byte count
             0000203D'EF    16    1715  9081         JSB     SEND_TU58                         ; send command packet
      24 00000000'EF    00    E1  171B  9082         BBC     #V_CLOCK_FLAG,B_FLAGS,190$ ; br = no timeout error
                                  1723  9083         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=NO_READY
             00001BEB'EF    DF    1723                       PUSHAL  NO_READY
             000018EB'EF    DF    1729                       PUSHAL  T_RD_BLK_ERR
                      00    DD    172F                       PUSHL   #0
                      07    DD    1731                       PUSHL   #$ER
                                  1733       ;;; TEST 12, SUBTEST 10, ERROR 7
         000100D0 9F    04    FB  1733                       CALLS   #$$M, @#DS$ERRHARD
                                  173A  9084         $DS_BNCOMPLETE  180$
                   03 50    E9    173A                       BLBC    R0, 180$
                                  173D  9085         $DS_ABORT       TEST
                      50    D4    173D                       CLRL    R0                        ; SEND A WARNING TO THE SUPERVISOR
                            04    173F                       RET                               ; TERMINATE TEST
                                  1740  9086 180$:   $DS_ABORT
         00010020 9F    6C    FA  1740                       CALLG   (AP), @#DS$ABORT
                                  1747  9087                                                   ; read data and end packets
         54  FFF3CB00 8F    D0    1747  9088 190$:   MOVL    #DELAY_TIME3,R4                   ; load receive packet delay time
         55  00000000'FF    9E    174E  9089         MOVAB   @A_BUFFER_BEGIN,R5                ; load buffer address to store data
     57  55  00000210 8F    C1    1755  9090         ADDL3   #528,R5,R7                        ; load last expected address of data
      07 00000001'EF    02    E1  175D  9091         BBC     #V_REPEAT_CHECK,B_FLAGS2,195$ ; br = not repeating
         57  00000210 8F    C0    1765  9092         ADDL2   #528,R7                           ; must double buffer if repeating
                   57 0E    C0    176C  9093 195$:   ADDL2   #14,R7                            ; make last address include end packet
             000020C8'EF    16    176F  9094         JSB     MRECEIVE_TU58                     ; go get data from TU58
         5B  00000000'FF    9E    1775  9095         MOVAB   @A_BUFFER_BEGIN,R11               ; load address of actual data
                   5A 93    DE    177C  9096         MOVAL   @(R3)+,R10                        ; load address of expected data
                   57 55    D1    177F  9097         CMPL    R5,R7                             ; check for correct number of bytes
                   31 13    1782        9098         BEQL    230$                              ; br = correct number of bytes
         00000000'EF    04    88  1784  9099         BISB2   #M_DATA_FLAG,B_FLAGS              ; set data error flag
                                  178B  9100 197$:   $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=REC_ERR,-
                                  178B  9101                         P1=#T_REC_DATA
             00001D68'8F    DD    178B                       PUSHL   #T_REC_DATA
             00001EB4'EF    DF    1791                       PUSHAL  REC_ERR
             000018EB'EF    DF    1797                       PUSHAL  T_RD_BLK_ERR
                      00    DD    179D                       PUSHL   #0
                      08    DD    179F                       PUSHL   #$ER
                                  17A1       ;;; TEST 12, SUBTEST 10, ERROR 8
         000100D0 9F    05    FB  17A1                       CALLS   #$$M, @#DS$ERRHARD
                                  17A8  9102         $DS_BNCOMPLETE  200$
                   03 50    E9    17A8                       BLBC    R0, 200$
                                  17AB  9103         $DS_ABORT       TEST
                      50    D4    17AB                       CLRL    R0                        ; SEND A WARNING TO THE SUPERVISOR
                            04    17AD                       RET                               ; TERMINATE TEST
                                  17AE  9104
                                  17AE  9105 200$:   $DS_ABORT
      00010020 9F    6C   FA  17AE                          CALLG   (AP), @#DS$ABORT
                              17B5  9106                                                   ; check data packets
            00000006'EF   D4  17B5  9107 230$:    CLRL      L_COUNT                        ; clear counter to check 4 packets
      5C    00000084 8F   D0  17BB  9108          MOVL      #132,R12                       ; load number of bytes to check
                              17C2  9109 240$:    $DS_BREAK
      00010058 9F    6E   FA  17C2                          CALLG   (SP), @#DS$BREAK
            0000213B'EF   16  17C9  9110          JSB       CHECK_PACKET                   ; go check data packet
                              17CF  9111          $DS_BNCOMPLETE  260$                     ; br = no error
                  24 50   E9  17CF                          BLBC    R0, 260$
                              17D2  9112          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=DATA_ERR_DATA
            00001E75'EF   DF  17D2                          PUSHAL  DATA_ERR_DATA
            000018EB'EF   DF  17D8                          PUSHAL  T_RD_BLK_ERR
                     00   DD  17DE                          PUSHL   #0
                     09   DD  17E0                          PUSHL   #$ER
                              17E2        ;;; TEST 12, SUBTEST 10, ERROR 9
      000100D0 9F    04   FB  17E2                          CALLS   #$$M, @#DS$ERRHARD
                              17E9  9113          $DS_BNCOMPLETE  250$
                  03 50   E9  17E9                          BLBC    R0, 250$
                              17EC  9114          $DS_ABORT       TEST
                     50   D4  17EC                          CLRL    R0                     ; SEND A WARNING TO THE SUPERVISOR
                          04  17EE                          RET                            ; TERMINATE TEST
                              17EF  9115 250$:    $DS_ABORT
      00010020 9F    6C   FA  17EF                          CALLG   (AP), @#DS$ABORT
      5B    00000084 8F   C0  17F6  9116 260$:    ADDL2     #132,R11                       ; adjust pointer to next packet
                  5A 93   DE  17FD  9117          MOVAL     @(R3)+,R10                     ; load address of expected data
   BA 00000006'EF    03   F3  1800  9118          AOBLEQ    #3,L_COUNT,240$                ; br = not finished checking data
   0F 00000001'EF    02   E1  1808  9119          BBC       #V_REPEAT_CHECK,B_FLAGS2,265$ ; br = not repeating
            00000006'EF   D4  1810  9120          CLRL      L_COUNT                        ; initialize check counter
            00000001'EF   94  1816  9121          CLRB      B_FLAGS2                       ; clear repeat flag to stop repeating
                   FFA3   31  181C  9122          BRW       240$                           ; check rest of data packets
                              181F  9123                                                   ; check end packet
                  5C 0E   D0  181F  9124 265$:    MOVL      #14,R12                        ; load number of bytes to check
                              1822  9125          $DS_BREAK
      00010058 9F    6E   FA  1822                          CALLG   (SP), @#DS$BREAK
            0000213B'EF   16  1829  9126          JSB       CHECK_PACKET                   ; go check end packet
                              182F  9127          $DS_BNCOMPLETE  275$                     ; br = no error
                  24 50   E9  182F                          BLBC    R0, 275$
                              1832  9128          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_BLK_ERR,PRLINK=DATA_ERR_END
            00001E9D'EF   DF  1832                          PUSHAL  DATA_ERR_END
            000018EB'EF   DF  1838                          PUSHAL  T_RD_BLK_ERR
                     00   DD  183E                          PUSHL   #0
                     0A   DD  1840                          PUSHL   #$ER
                              1842        ;;; TEST 12, SUBTEST 10, ERROR 10
      000100D0 9F    04   FB  1842                          CALLS   #$$M, @#DS$ERRHARD
                              1849  9129          $DS_BNCOMPLETE  270$
                  03 50   E9  1849                          BLBC    R0, 270$
                              184C  9130          $DS_ABORT       TEST
                     50   D4  184C                          CLRL    R0                     ; SEND A WARNING TO THE SUPERVISOR
                          04  184E                          RET                            ; TERMINATE TEST
                              184F  9131 270$:    $DS_ABORT
      00010020 9F    6C   FA  184F                          CALLG   (AP), @#DS$ABORT
 FCA9 00000004'EF    01   04  9D  1856  9132 275$:    ACBB      #4,#1,B_COUNT1,10$     ; br = not finished executing test
                              1860  9133
                              1860  9134
                              1860  9135          $DS_ENDSUB
```

```
                                   1860        T12_S10_X:
    00010038 9F   0000015C'EF   FA 1860                        CALLG   $S$, a#DS$ENDSUB
                                   186B  9136
                                   186B  9137          $DS_PAGE
```

```
186B               .SBTTL  SUBTEST 12.11:   RECORD WRITE - READ
186B   ;+
186B   ; SUBTEST DESCRIPTION:
186B   ;
186B   ;           This subtest sends the TU58 controller a series of command packets
186B   ;           that will cause head switches, forward seeks, reverse seeks and
186B   ;           seeks with head switches durring read and write operations on
186B   ;           records. All data written is read and checked. All end packets
186B   ;           are checked as the subtest progresses. If any errors occur they are
186B   ;           reported.
186B   ;
186B   ;
186B   ; SUBTEST STEPS:
186B   ;
186B   ;           SUBTEST RECORD WRITE - READ
186B   ;           :  DO for count = 1 to 5
186B   ;           :  :  Point to write command packet
186B   ;           :  :  Clear buffer
186B   ;           :  :  REPEAT
186B   ;           :  :  :  Send command packet byte at a time
186B   ;           :  :  :  Set timer
186B   ;           :  :  :  IF timer expired AND no response
186B   ;           :  :  :  :  THEN
186B   ;           :  :  :  :  Report no response error
186B   ;           :  :  :  :  Exit test
186B   ;           :  :  :  ENDIF
186B   ;           :  :  : UNTIL all bytes sent
186B   ;           :  :  ENDREPEAT
186B   ;           :  :  DO for count = 1 to 2
186B   ;           :  :  :  Wait for continue
186B   ;           :  :  :  Set timer
186B   ;           :  :  :  IF timer expired AND no continue
186B   ;           :  :  :  :  THEN
186B   ;           :  :  :  :  Report no continue error
186B   ;           :  :  :  :  Exit test
186B   ;           :  :  :  ENDIF
186B   ;           :  :  :  Point to data packet
186B   ;           :  :  :  REPEAT
186B   ;           :  :  :  :  Send data packet byte at a time
186B   ;           :  :  :  :  Set timer
186B   ;           :  :  :  :  IF timer expired AND no response
186B   ;           :  :  :  :  :  THEN
186B   ;           :  :  :  :  :  Report no response error
186B   ;           :  :  :  :  :  Exit test
186B   ;           :  :  :  :  ENDIF
186B   ;           :  :  :  : UNTIL all bytes sent
186B   ;           :  :  :  ENDREPEAT
186B   ;           :  :  ENDDO
186B   ;           :  :  REPEAT
186B   ;           :  :  :  Set timer
186B   ;           :  :  :  IF timer expired AND done set
186B   ;           :  :  :  :  THEN
186B   ;           :  :  :  :  :  Read byte
186B   ;           :  :  :  ENDIF
186B   ;           :  :  : UNTIL timer expired AND done clear
```

```
186B  ;        :  :  ENDREPEAT
186B  ;        :  :  IF expected byte count NEQ actual byte count
186B  ;        :  :  :  THEN
186B  ;        :  :  :  :  Report error
186B  ;        :  :  :  :  Exit test
186B  ;        :  :  ENDIF
186B  ;        :  :  REPEAT
186B  ;        :  :  :  Point to expected end packet
186B  ;        :  :  :  IF expected end data NEQ received end data
186B  ;        :  :  :  :  THEN
186B  ;        :  :  :  :  Report end packet data error
186B  ;        :  :  :  ENDIF
186B  ;        :  :  : UNTIL all data checked
186B  ;        :  :  ENDREPEAT
186B  ;        :  :  Point to read command packet
186B  ;        :  :  REPEAT
186B  ;        :  :  :  Send command packet byte at a time
186B  ;        :  :  :  Set timer
186B  ;        :  :  :  IF timer expired and no response
186B  ;        :  :  :  :  THEN
186B  ;        :  :  :  :  Report no response error
186B  ;        :  :  :  :  Exit test
186B  ;        :  :  :  ENDIF
186B  ;        :  :  : UNTIL all bytes sent
186B  ;        :  :  ENDREPEAT
186B  ;        :  :  REPEAT
186B  ;        :  :  :  Set timer
186B  ;        :  :  :  IF timer expired AND done set
186B  ;        :  :  :  :  THEN
186B  ;        :  :  :  :  :  Read byte
186B  ;        :  :  :  ENDIF
186B  ;        :  :  : UNTIL timer expired AND done clear
186B  ;        :  :  ENDREPEAT
186B  ;        :  :  IF expected byte count NEQ actual byte count
186B  ;        :  :  :  THEN
186B  ;        :  :  :  :  Report error
186B  ;        :  :  :  :  Exit test
186B  ;        :  :  ENDIF
186B  ;        :  :  REPEAT
186B  ;        :  :  :  Point to expected data packet
186B  ;        :  :  :  REPEAT
186B  ;        :  :  :  :  IF data expected NEQ data received
186B  ;        :  :  :  :  :  THEN
186B  ;        :  :  :  :  :  Report data error
186B  ;        :  :  :  :  :  Exit subtest
186B  ;        :  :  :  :  ENDIF
186B  ;        :  :  :  :  Point to next byte of data
186B  ;        :  :  :  : UNTIL all data in packet checked
186B  ;        :  :  :  ENDREPEAT
186B  ;        :  :  : UNTIL 4 data packets checked
186B  ;        :  :  ENDREPEAT
186B  ;        :  :  Point to expected end packet
186B  ;        :  :  REPEAT
186B  ;        :  :  :  IF data expected NEQ data received
186B  ;        :  :  :  :  THEN
```

```
                                     186B  9249 ;       :  :  :  :  Report data error
                                     186B  9250 ;       :  :  :  :  Exit subtest
                                     186B  9251 ;       :  :  :  ENDIF
                                     186B  9252 ;       :  :  : UNTIL all end packet bytes checked
                                     186B  9253 ;       :  :  ENDREPEAT
                                     186B  9254 ;       :  ENDDO
                                     186B  9255 ;       ENDSUBTEST RECORD WRITE - READ
                                     186B  9256 ;
                                     186B  9257 ;
                                     186B  9258 ;.page
                                     186B  9259 ;-
                                     186B  9260         $DS_BGNSUB
                                     186B       T12_S11::
     00010030 9F    00000168'EF   FA  186B                      CALLG   $S$, @#DS$BGNSUB
                   00000001'EF   94  1876  9261         CLRB    B_FLAGS2                ; initialize repeat flag
              00000001'EF   06   88  187C  9262         BISB2   #<M_REPEAT_SEND+M_REPEAT_CHECK>,B_FLAGS2
                   00000004'EF   94  1883  9263         CLRB    B_COUNT1                ; initialize command counter
              53   0000176B'EF   9E  1889  9264         MOVAB   MCOMMAND_TBL_4,R3       ; load pointer to command table
                                     1890  9265                                         ; clear buffers
                   00000000'EF   94  1890  9266 10$:    CLRB    B_FLAGS                 ; initialize flags
              58   00000000'FF   9E  1896  9267         MOVAB   @A_BUFFER_BEGIN,R8      ; load buffer starting address
              59   00000600 8F   D0  189D  9268         MOVL    #BUFFER_SIZE,R9         ; do all pages
                   00002134'EF   16  18A4  9269         JSB     CLEAR_BUFFER            ; go clear buffers
                                     18AA  9270                                         ; send write command
              54   FFF3CB00 8F   D0  18AA  9271         MOVL    #DELAY_TIME3,R4         ; load send packet delay time
                      55    93   DE  18B1  9272         MOVAL   @(R3)+,R5               ; point to write command packet
                      57    0E   D0  18B4  9273         MOVL    #14,R7                  ; load transfer byte count
                   0000203D'EF   16  18B7  9274         JSB     SEND_TU58               ; send command packet
           24 00000000'EF   00   E1  18BD  9275         BBC     #V_CLOCK_FLAG,B_FLAGS,30$ ; br = no timeout error
                                     18C5  9276         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=NO_READY
                   00001BEB'EF   DF  18C5                       PUSHAL  NO_READY
                   0000190D'EF   DF  18CB                       PUSHAL  T_WR_REC_ERR
                            00   DD  18D1                       PUSHL   #0
                            01   DD  18D3                       PUSHL   #$ER
                                     18D5       ;;; TEST 12, SUBTEST 11, ERROR 1
              000100D0 9F   04   FB  18D5                       CALLS   #$$M, @#DS$ERRHARD
                                     18DC  9277         $DS_BNCOMPLETE  20$
                         03 50   E9  18DC                       BLBC    R0, 20$
                                     18DF  9278         $DS_ABORT       TEST
                            50   D4  18DF                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                            04   18E1                           RET                     ; TERMINATE TEST
                                     18E2  9279 20$:    $DS_ABORT
              00010020 9F   6C   FA  18E2                       CALLG   (AP), @#DS$ABORT
                                     18E9  9280                                         ; read continue
              54   FF7E4CE0 8F   D0  18E9  9281 30$:    MOVL    #DELAY_TIME7,R4         ; load receive packet delay time
              55   00000000'FF   9E  18F0  9282         MOVAB   @A_BUFFER_BEGIN,R5      ; load buffer address to store data
                  57   55   01   C1  18F7  9283         ADDL3   #1,R5,R7                ; load last expected address of data
                   000020C8'EF   16  18FB  9284         JSB     MRECEIVE_TU58           ; go get data from TU58
              5A   00000023'EF   9E  1901  9285         MOVAB   B_CONTINUE,R10          ; load address of expected data
              5B   00000000'FF   9E  1908  9286         MOVAB   @A_BUFFER_BEGIN,R11     ; load address of actual data
                       57   55   D1  190F  9287         CMPL    R5,R7                   ; check for correct number of bytes
                            2A   13  1912  9288         BEQL    70$                     ; br = correct number of bytes
                                     1914  9289         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=REC_ERR,-
                                     1914  9290                         P1=#T_REC_CONT
                   00001D47'8F   DD  1914                       PUSHL   #T_REC_CONT
```

```
          00001EB4'EF    DF  191A                    PUSHAL  REC_ERR
          0000190D'EF    DF  1920                    PUSHAL  T_WR_REC_ERR
                   00    DD  1926                    PUSHL   #0
                   02    DD  1928                    PUSHL   #$ER
                             192A          ;;; TEST 12, SUBTEST 11, ERROR 2
      000100D0 9F    05  FB  192A                    CALLS   #$$M, @#DS$ERRHARD
                             1931  9291          $DS_BNCOMPLETE  40$
                03 50    E9  1931                    BLBC    R0, 40$
                             1934  9292          $DS_ABORT       TEST
                   50    D4  1934                    CLRL    R0                    ; SEND A WARNING TO THE SUPERVISOR
                   04        1936                    RET                           ; TERMINATE TEST
                             1937  9293
                             1937  9294 40$:     $DS_ABORT
      00010020 9F    6C  FA  1937                    CALLG   (AP), @#DS$ABORT
                             193E  9295                                            ; check continue
             5C    01    D0  193E  9296 70$:     MOVL    #1,R12                    ; load number of bytes to check
                             1941  9297          $DS_BREAK
      00010058 9F    6E  FA  1941                    CALLG   (SP), @#DS$BREAK
          0000213B'EF    16  1948  9298          JSB     CHECK_PACKET              ; go check continue
                             194E  9299          $DS_BNCOMPLETE  90$               ; br = no error
                24 50    E9  194E                    BLBC    R0, 90$
                             1951  9300          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=DATA_ERR_CONT
          00001E4D'EF    DF  1951                    PUSHAL  DATA_ERR_CONT
          0000190D'EF    DF  1957                    PUSHAL  T_WR_REC_ERR
                   00    DD  195D                    PUSHL   #0
                   03    DD  195F                    PUSHL   #$ER
                             1961          ;;; TEST 12, SUBTEST 11, ERROR 3
      000100D0 9F    04  FB  1961                    CALLS   #$$M, @#DS$ERRHARD
                             1968  9301          $DS_BNCOMPLETE  80$
                03 50    E9  1968                    BLBC    R0, 80$
                             196B  9302          $DS_ABORT       TEST
                   50    D4  196B                    CLRL    R0                    ; SEND A WARNING TO THE SUPERVISOR
                   04        196D                    RET                           ; TERMINATE TEST
                             196E  9303 80$:     $DS_ABORT
      00010020 9F    6C  FA  196E                    CALLG   (AP), @#DS$ABORT
                             1975  9304                                            ; send data packet
   54     FFFB6C20 8F    D0  1975  9305 90$:     MOVL    #DELAY_TIME1,R4           ; load send packet delay time
               55    93  DE  197C  9306          MOVAL   @(R3)+,R5                 ; point to data packet
   57     00000084 8F    D0  197F  9307          MOVL    #132,R7                   ; load transfer byte count
          0000203D'EF    16  1986  9308          JSB     SEND_TU58                 ; send data packet
   24 00000000'EF    00  E1  198C  9309          BBC     #V_CLOCK_FLAG,B_FLAGS,110$ ; br = no timeout error
                             1994  9310          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=NO_READY
          00001BEB'EF    DF  1994                    PUSHAL  NO_READY
          0000190D'EF    DF  199A                    PUSHAL  T_WR_REC_ERR
                   00    DD  19A0                    PUSHL   #0
                   04    DD  19A2                    PUSHL   #$ER
                             19A4          ;;; TEST 12, SUBTEST 11, ERROR 4
      000100D0 9F    04  FB  19A4                    CALLS   #$$M, @#DS$ERRHARD
                             19AB  9311          $DS_BNCOMPLETE  100$
                03 50    E9  19AB                    BLBC    R0, 100$
                             19AE  9312          $DS_ABORT       TEST
                   50    D4  19AE                    CLRL    R0                    ; SEND A WARNING TO THE SUPERVISOR
                   04        19B0                    RET                           ; TERMINATE TEST
                             19B1  9313 100$:    $DS_ABORT
      00010020 9F    6C  FA  19B1                    CALLG   (AP), @#DS$ABORT
```

```
                                19B8  9314
          00000000'FF    94     19B8  9315 110$:   CLRB    @A_BUFFER_BEGIN         ; clear buffer used to store continue
      0A 00000001'EF     01     E1    19BE  9316   BBC     #V_REPEAT_SEND,B_FLAGS2,115$ ; br =  not repeating
         00000001'EF     02     8A    19C6  9317   BICB2   #M_REPEAT_SEND,B_FLAGS2 ; clear repeat flag to stop repeating
                     FF19  31   19CD  9318         BRW     30$                     ; finish sending data
                                19D0  9319                                         ; read end packet
      54    FF7E4CE0 8F   D0    19D0  9320 115$:   MOVL    #DELAY_TIME7,R4         ; load receive packet delay time
      55    00000000'FF   9E    19D7  9321         MOVAB   @A_BUFFER_BEGIN,R5      ; load buffer address to store data
          57    55   0E   C1    19DE  9322         ADDL3   #14,R5,R7               ; load last expected address of data
          000020C8'EF     16    19E2  9323         JSB     MRECEIVE_TU58           ; go get data from TU58
                5A   93   DE    19E8  9324         MOVAL   @(R3)+,R10              ; load address of expected end packet
      5B    00000000'FF   9E    19EB  9325         MOVAB   @A_BUFFER_BEGIN,R11     ; load address of actual end packet
                57   55   D1    19F2  9326         CMPL    R5,R7                   ; check for correct number of bytes
                     31   13    19F5  9327         BEQL    150$                    ; br = correct number of bytes
         00000000'EF    08  88  19F7  9328         BISB2   #M_END_FLAG,B_FLAGS     ; set end error flag
                                19FE  9329 117$:   $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=REC_ERR,-
                                19FE  9330                          P1=#T_REC_END
          00001D2E'8F     DD    19FE                        PUSHL   #T_REC_END
          00001EB4'EF     DF    1A04                        PUSHAL  REC_ERR
          0000190D'EF     DF    1A0A                        PUSHAL  T_WR_REC_ERR
                     00   DD    1A10                        PUSHL   #0
                     05   DD    1A12                        PUSHL   #$ER
                                1A14       ;;; TEST 12, SUBTEST 11, ERROR 5
         000100D0 9F     05   FB    1A14                    CALLS   #$$M, @#DS$ERRHARD
                                1A1B  9331         $DS_BNCOMPLETE  120$
                 03 50    E9    1A1B                        BLBC    R0, 120$
                                1A1E  9332         $DS_ABORT       TEST
                     50   D4    1A1E                        CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                          04    1A20                        RET                     ; TERMINATE TEST
                                1A21  9333
                                1A21  9334 120$:   $DS_ABORT
         00010020 9F     6C   FA    1A21                    CALLG   (AP), @#DS$ABORT
                                1A28  9335                                         ; check end packet
                5C   0E   D0    1A28  9336 150$:   MOVL    #14,R12                 ; load number of bytes to check
                                1A2B  9337         $DS_BREAK
         00010058 9F     6E   FA    1A2B                    CALLG   (SP), @#DS$BREAK
          0000213B'EF     16    1A32  9338         JSB     CHECK_PACKET            ; go check end packet
                                1A38  9339         $DS_BNCOMPLETE  160$            ; br = no error
                 21 50    E9    1A38                        BLBC    R0, 160$
                                1A3B  9340         $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_REC_ERR,PRLINK=DATA_ERR_END
          00001E9D'EF     DF    1A3B                        PUSHAL  DATA_ERR_END
          0000190D'EF     DF    1A41                        PUSHAL  T_WR_REC_ERR
                     00   DD    1A47                        PUSHL   #0
                     06   DD    1A49                        PUSHL   #$ER
                                1A4B       ;;; TEST 12, SUBTEST 11, ERROR 6
         000100D0 9F     04   FB    1A4B                    CALLS   #$$M, @#DS$ERRHARD
                                1A52  9341         $DS_BCOMPLETE   160$            ; br = error, but continue
                 07 50    E8    1A52                        BLBS    R0, 160$
                                1A55  9342         $DS_ABORT
         00010020 9F     6C   FA    1A55                    CALLG   (AP), @#DS$ABORT
                                1A5C  9343                                         ; clear buffers
          00000000'EF     94    1A5C  9344 160$:   CLRB    B_FLAGS                 ; initialize flags
      58    00000000'FF   9E    1A62  9345         MOVAB   @A_BUFFER_BEGIN,R8      ; load buffer starting address
      59    00000600 8F   D0    1A69  9346         MOVL    #BUFFER_SIZE,R9         ; do all pages
          00002134'EF     16    1A70  9347         JSB     CLEAR_BUFFER            ; go clear buffers
```

```
                                      1A76  9348                                           ; send read command
              54    FFFB6C20 8F   D0  1A76  9349          MOVL    #DELAY_TIME1,R4          ; load send packet delay time
                        55   93   DE  1A7D  9350          MOVAL   @(R3)+,R5                ; point to read command packet
                        57   0E   D0  1A80  9351          MOVL    #14,R7                   ; load transfer byte count
                    0000203D'EF   16  1A83  9352          JSB     SEND_TU58                ; send command packet
           24 00000000'EF    00   E1  1A89  9353          BBC     #V_CLOCK_FLAG,B_FLAGS,190$ ; br = no timeout error
                                      1A91  9354          $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=NO_READY
                    00001BEB'EF   DF  1A91                        PUSHAL  NO_READY
                    00001931'EF   DF  1A97                        PUSHAL  T_RD_REC_ERR
                             00   DD  1A9D                        PUSHL   #0
                             07   DD  1A9F                        PUSHL   #$ER
                                      1AA1       ;;; TEST 12, SUBTEST 11, ERROR 7
              000100D0 9F    04   FB  1AA1                        CALLS   #$$M, @#DS$ERRHARD
                                      1AA8  9355          $DS_BNCOMPLETE  180$
                          03 50   E9  1AA8                        BLBC    R0, 180$
                                      1AAB  9356          $DS_ABORT       TEST
                             50   D4  1AAB                        CLRL    R0               ; SEND A WARNING TO THE SUPERVISOR
                                  04  1AAD                        RET                      ; TERMINATE TEST
                                      1AAE  9357 180$:    $DS_ABORT
              00010020 9F    6C   FA  1AAE                        CALLG   (AP), @#DS$ABORT
                                      1AB5  9358                                           ; read data and end packets
              54    FFF3CB00 8F   D0  1AB5  9359 190$:    MOVL    #DELAY_TIME3,R4          ; load receive packet delay time
              55    00000000'FF   9E  1ABC  9360          MOVAB   @A_BUFFER_BEGIN,R5       ; load buffer address to store data
         57   55    00000084 8F   C1  1AC3  9361          ADDL3   #132,R5,R7               ; load last expected address of data
           07 00000001'EF    02   E1  1ACB  9362          BBC     #V_REPEAT_CHECK,B_FLAGS2,195$ ; br = not repeating
              57    00000084 8F   C0  1AD3  9363          ADDL2   #132,R7                  ; must double buffer if repeating
                        57   0E   C0  1ADA  9364 195$:    ADDL2   #14,R7                   ; add end packet to address
                    000020C8'EF   16  1ADD  9365          JSB     MRECEIVE_TU58            ; go get data from TU58
              5B    00000000'FF   9E  1AE3  9366          MOVAB   @A_BUFFER_BEGIN,R11      ; load address of buffer
                        5A   93   DE  1AEA  9367          MOVAL   @(R3)+,R10               ; load address of expected data
                        57   55   D1  1AED  9368          CMPL    R5,R7                    ; check for correct number of bytes
                             31   13  1AF0  9369          BEQL    230$                     ; br = correct number of bytes
              00000000'EF    04   88  1AF2  9370          BISB2   #M_DATA_FLAG,B_FLAGS     ; set data error flag
                                      1AF9  9371 197$:    $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=REC_ERR,-
                                      1AF9  9372                          P1=#T_REC_DATA
                    00001D68'8F   DD  1AF9                        PUSHL   #T_REC_DATA
                    00001EB4'EF   DF  1AFF                        PUSHAL  REC_ERR
                    00001931'EF   DF  1B05                        PUSHAL  T_RD_REC_ERR
                             00   DD  1B0B                        PUSHL   #0
                             08   DD  1B0D                        PUSHL   #$ER
                                      1B0F       ;;; TEST 12, SUBTEST 11, ERROR 8
              000100D0 9F    05   FB  1B0F                        CALLS   #$$M, @#DS$ERRHARD
                                      1B16  9373          $DS_BNCOMPLETE  200$
                          03 50   E9  1B16                        BLBC    R0, 200$
                                      1B19  9374          $DS_ABORT       TEST
                             50   D4  1B19                        CLRL    R0               ; SEND A WARNING TO THE SUPERVISOR
                                  04  1B1B                        RET                      ; TERMINATE TEST
                                      1B1C  9375
                                      1B1C  9376 200$:    $DS_ABORT
              00010020 9F    6C   FA  1B1C                        CALLG   (AP), @#DS$ABORT
                                      1B23  9377                                           ; check data packets
              5C    00000084 8F   D0  1B23  9378 230$:    MOVL    #132,R12                 ; load number of bytes to check
                                      1B2A  9379 240$:    $DS_BREAK
              00010058 9F    6E   FA  1B2A                        CALLG   (SP), @#DS$BREAK
                    0000213B'EF   16  1B31  9380          JSB     CHECK_PACKET             ; go check data packet
```

```
                               1B37  9381         $DS_BNCOMPLETE  260$          ; br = no error
                   24 50   E9  1B37                       BLBC    R0, 260$
                               1B3A  9382         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=DATA_ERR_DATA
             00001E75'EF   DF  1B3A                       PUSHAL  DATA_ERR_DATA
             00001931'EF   DF  1B40                       PUSHAL  T_RD_REC_ERR
                      00   DD  1B46                       PUSHL   #0
                      09   DD  1B48                       PUSHL   #$ER
                               1B4A       ;;; TEST 12, SUBTEST 11, ERROR 9
       000100D0 9F    04   FB  1B4A                       CALLS   #$$M, @#DS$ERRHARD
                               1B51  9383         $DS_BNCOMPLETE  250$
                   03 50   E9  1B51                       BLBC    R0, 250$
                               1B54  9384         $DS_ABORT       TEST
                      50   D4  1B54                       CLRL    R0            ; SEND A WARNING TO THE SUPERVISOR
                           04  1B56                       RET                   ; TERMINATE TEST
                               1B57  9385 250$:   $DS_ABORT
       00010020 9F    6C   FA  1B57                       CALLG   (AP), @#DS$ABORT
       5B    00000084 8F   C0  1B5E  9386 260$:   ADDL2   #132,R11              ; adjust pointer to next packet
    0C 00000001'EF    02   E1  1B65  9387         BBC     #V_REPEAT_CHECK,B_FLAGS2,265$ ; br = not repeating
             00000001'EF   94  1B6D  9388         CLRB    B_FLAGS2              ; clear repeat flag to stop repeating
                5A    93   DE  1B73  9389         MOVAL   @(R3)+,R10            ; load address of expected data
                    FFB1   31  1B76  9390         BRW     240$                  ; check rest of data packets
                               1B79  9391                                       ; check end packet
                5A    93   DE  1B79  9392 265$:   MOVAL   @(R3)+,R10            ; load address of expected end packet
                5C    0E   D0  1B7C  9393         MOVL    #14,R12               ; load number of bytes to check
                               1B7F  9394         $DS_BREAK
       00010058 9F    6E   FA  1B7F                       CALLG   (SP), @#DS$BREAK
             0000213B'EF   16  1B86  9395         JSB     CHECK_PACKET          ; go check end packet
                               1B8C  9396         $DS_BNCOMPLETE  275$          ; br = no error
                   24 50   E9  1B8C                       BLBC    R0, 275$
                               1B8F  9397         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_RD_REC_ERR,PRLINK=DATA_ERR_END
             00001E9D'EF   DF  1B8F                       PUSHAL  DATA_ERR_END
             00001931'EF   DF  1B95                       PUSHAL  T_RD_REC_ERR
                      00   DD  1B9B                       PUSHL   #0
                      0A   DD  1B9D                       PUSHL   #$ER
                               1B9F       ;;; TEST 12, SUBTEST 11, ERROR 10
       000100D0 9F    04   FB  1B9F                       CALLS   #$$M, @#DS$ERRHARD
                               1BA6  9398         $DS_BNCOMPLETE  270$
                   03 50   E9  1BA6                       BLBC    R0, 270$
                               1BA9  9399         $DS_ABORT       TEST
                      50   D4  1BA9                       CLRL    R0            ; SEND A WARNING TO THE SUPERVISOR
                           04  1BAB                       RET                   ; TERMINATE TEST
                               1BAC  9400 270$:   $DS_ABORT
       00010020 9F    6C   FA  1BAC                       CALLG   (AP), @#DS$ABORT
FCD3 00000004'EF  01  04   9D  1BB3  9401 275$:   ACBB    #4,#1,B_COUNT1,10$    ; br = not finished executing test
                               1BBD  9402
                               1BBD  9403
                               1BBD  9404         $DS_ENDSUB
                               1BBD       T12_S11_X:
00010038 9F  00000168'EF   FA  1BBD                       CALLG   $S$, @#DS$ENDSUB
                               1BC8  9405         $DS_PAGE
```

```
1BC8              .SBTTL  SUBTEST 12.12:   WRITE RECORD INTERRUPT
1BC8  9408 ;+
1BC8  9409 ; SUBTEST DESCRIPTION:
1BC8  9410 ;
1BC8  9411 ;          This subtest sends the TU58 a command packet to write one record
1BC8  9412 ;          of 128 bytes in interrupt mode. When an interrupt occurs the write service-
1BC8  9413 ;          ice routine sends the next byte. A timeout occurs after sufficient time
1BC8  9414 ;          has passed to send a full packet, at which time an end packet is
1BC8  9415 ;          read and checked. An error is reported if an interrupt fails to occur
1BC8  9416 ;          before the timer expires and the full packet has not been sent. The data
1BC8  9417 ;          packet sent contains alternating bytes of all one's and all zero's.
1BC8  9418 ;          The data will be read and checked for errors in the next subtest.
1BC8  9419 ;
1BC8  9420 ;
1BC8  9421 ; SUBTEST STEPS:
1BC8  9422 ;
1BC8  9423 ;          SUBTEST WRITE RECORD INTERRUPT
1BC8  9424 ;          :  Clear buffer
1BC8  9425 ;          :  Point to write command packet
1BC8  9426 ;          :  REPEAT
1BC8  9427 ;          :  :  Send command packet byte at a time
1BC8  9428 ;          :  :  Set timer
1BC8  9429 ;          :  :  IF timer expired AND no response
1BC8  9430 ;          :  :  :  THEN
1BC8  9431 ;          :  :  :  Report no response error
1BC8  9432 ;          :  :  :  Exit test
1BC8  9433 ;          :  :  ENDIF
1BC8  9434 ;          :  : UNTIL all bytes sent
1BC8  9435 ;          :  ENDREPEAT
1BC8  9436 ;          :  Wait for continue
1BC8  9437 ;          :  Set timer
1BC8  9438 ;          :  IF timer expired AND no continue
1BC8  9439 ;          :  :  THEN
1BC8  9440 ;          :  :  Report no continue error
1BC8  9441 ;          :  :  Exit test
1BC8  9442 ;          :  ENDIF
1BC8  9443 ;          :  Point to data packet
1BC8  9444 ;          :  Enable send interrupt
1BC8  9445 ;          :  REPEAT
1BC8  9446 ;          :  :  Send data packet byte at a time
1BC8  9447 ;          :  :  Set timer
1BC8  9448 ;          :  :  IF timer expired AND no response
1BC8  9449 ;          :  :  :  THEN
1BC8  9450 ;          :  :  :  :  DISABLE send interrupt
1BC8  9451 ;          :  :  :  :  Report no response error
1BC8  9452 ;          :  :  :  :  Exit test
1BC8  9453 ;          :  :  ENDIF
1BC8  9454 ;          :  : UNTIL all bytes sent
1BC8  9455 ;          :  ENDREPEAT
1BC8  9456 ;          :  DISABLE send interrupt
1BC8  9457 ;          :  REPEAT
1BC8  9458 ;          :  :  Set timer
1BC8  9459 ;          :  :  IF timer expired AND done set
1BC8  9460 ;          :  :  :  THEN
1BC8  9461 ;          :  :  :  :  Read byte
```

```
                                      1BC8  9462 ;         :  :  ENDIF
                                      1BC8  9463 ;         :  : UNTIL timer expired AND done clear
                                      1BC8  9464 ;         :  ENDREPEAT
                                      1BC8  9465 ;         :  IF expected byte count NEQ actual byte count
                                      1BC8  9466 ;         :  :  THEN
                                      1BC8  9467 ;         :  :  :  Report error
                                      1BC8  9468 ;         :  :  :  Exit test
                                      1BC8  9469 ;         :  ENDIF
                                      1BC8  9470 ;         :  REPEAT
                                      1BC8  9471 ;         :  :  Point to expected end packet
                                      1BC8  9472 ;         :  :  IF expected end packet data NEQ received end packet data
                                      1BC8  9473 ;         :  :  :  THEN
                                      1BC8  9474 ;         :  :  :  Report end packet data error
                                      1BC8  9475 ;         :  :  ENDIF
                                      1BC8  9476 ;         :  : UNTIL all data checked
                                      1BC8  9477 ;         :  ENDREPEAT
                                      1BC8  9478 ;         ENDSUBTEST WRITE RECORD INTERRUPT
                                      1BC8  9479 ;
                                      1BC8  9480 ;
                                      1BC8  9481 ;
                                      1BC8  9482 ;.page
                                      1BC8  9483 ;-
                                      1BC8  9484
                                      1BC8  9485         $DS_BGNSUB
                                      1BC8       T12_S12::
       00010030 9F    00000174'EF   FA 1BC8                       CALLG    $S$, @#DS$BGNSUB
                                      1BD3  9486         $DS_SETVEC_S     VECTOR=#SCB$L_CSTD,SRVADR=WRITE_SERVICE
                              00   DD 1BD3                        PUSHL    #0
                    000021D4'EF   DF 1BD5                        PUSHAL   WRITE_SERVICE
                    000000F4 8F   DD 1BDB                        PUSHL    #SCB$L_CSTD
              00010160 9F    03   FB 1BE1                        CALLS    #3, @#DS$SETVEC
                    00000000'EF   94 1BE8  9487         CLRB       B_FLAGS                   ; initialize flags
                    00000002'EF   94 1BEE  9488         CLRB       B_FLAGS3
                                      1BF4  9489                                             ; clear buffers
              58    00000000'FF   9E 1BF4  9490         MOVAB      @A_BUFFER_BEGIN,R8        ; load buffer starting address
              59    00000400 8F   D0 1BFB  9491         MOVL       #1024,R9                  ; do two pages
                    00002134'EF   16 1C02  9492         JSB        CLEAR_BUFFER              ; go clear buffers
                                      1C08  9493                                             ; send write command
              54    FFE17B80 8F   D0 1C08  9494         MOVL       #DELAY_TIME5,R4           ; load send packet delay time
              55    0000035E'EF   9E 1C0F  9495         MOVAB      MB_WRITE_3_CMD,R5         ; point to write command packet
                          57  0E   D0 1C16  9496         MOVL       #14,R7                    ; load transfer byte count
                    0000203D'EF   16 1C19  9497         JSB        SEND_TU58                 ; send command packet
          24 00000000'EF    00   E1 1C1F  9498         BBC        #V_CLOCK_FLAG,B_FLAGS,20$ ; br = no timeout error
                                      1C27  9499         $DS_ERRHARD_S     ,UNIT=#0,MSGADR=T_WR_INT_ERR,PRLINK=NO_READY
                    00001BEB'EF   DF 1C27                        PUSHAL   NO_READY
                    000019D1'EF   DF 1C2D                        PUSHAL   T_WR_INT_ERR
                              00   DD 1C33                        PUSHL    #0
                              01   DD 1C35                        PUSHL    #$ER
                                      1C37       ;;; TEST 12, SUBTEST 12, ERROR 1
              000100D0 9F    04   FB 1C37                        CALLS    #$$M, @#DS$ERRHARD
                                      1C3E  9500         $DS_BNCOMPLETE   10$
                         03 50   E9 1C3E                        BLBC     R0, 10$
                                      1C41  9501         $DS_ABORT        TEST
                              50   D4 1C41                        CLRL     R0                  ; SEND A WARNING TO THE SUPERVISOR
                              04      1C43                        RET                          ; TERMINATE TEST
```

```
                                     1C44  9502 10$:    $DS_ABORT
              00010020 9F   6C   FA  1C44                       CALLG   (AP), @#DS$ABORT
                                     1C4B  9503                                         ; read continue
              54    FFF3CB00 8F    D0  1C4B  9504 20$:    MOVL    #DELAY_TIME3,R4         ; load receive packet delay time
              55    00000000'FF    9E  1C52  9505         MOVAB   @A_BUFFER_BEGIN,R5      ; load buffer address to store data
                  57   55   01    C1  1C59  9506         ADDL3   #1,R5,R7                ; load last expected address of data
                   000020C8'EF    16  1C5D  9507         JSB     MRECEIVE_TU58           ; go get data from TU58
              5A    00000023'EF    9E  1C63  9508         MOVAB   B_CONTINUE,R10          ; load address of expected data
              5B    00000000'FF    9E  1C6A  9509         MOVAB   @A_BUFFER_BEGIN,R11     ; load address of actual data
                       57   55    D1  1C71  9510         CMPL    R5,R7                   ; check for correct number of bytes
                            2A    13  1C74  9511         BEQL    60$                     ; br = correct number of bytes
                                     1C76  9512 25$:    $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_INT_ERR,PRLINK=REC_ERR,-
                                     1C76  9513                         P1=#T_REC_CONT
                   00001D47'8F    DD  1C76                       PUSHL   #T_REC_CONT
                   00001EB4'EF    DF  1C7C                       PUSHAL  REC_ERR
                   000019D1'EF    DF  1C82                       PUSHAL  T_WR_INT_ERR
                            00    DD  1C88                       PUSHL   #0
                            02    DD  1C8A                       PUSHL   #$ER
                                     1C8C       ;;; TEST 12, SUBTEST 12, ERROR 2
              000100D0 9F   05   FB  1C8C                       CALLS   #$$M, @#DS$ERRHARD
                                     1C93  9514         $DS_BNCOMPLETE  30$
                         03 50    E9  1C93                       BLBC    R0, 30$
                                     1C96  9515         $DS_ABORT       TEST
                            50    D4  1C96                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                  04  1C98                       RET                     ; TERMINATE TEST
                                     1C99  9516
                                     1C99  9517 30$:    $DS_ABORT
              00010020 9F   6C   FA  1C99                       CALLG   (AP), @#DS$ABORT
                                     1CA0  9518                                         ; check continue
                       5C   01    D0  1CA0  9519 60$:    MOVL    #1,R12                  ; load number of bytes to check
                                     1CA3  9520         $DS_BREAK
              00010058 9F   6E   FA  1CA3                       CALLG   (SP), @#DS$BREAK
                   0000213B'EF    16  1CAA  9521         JSB     CHECK_PACKET            ; go check continue
                                     1CB0  9522         $DS_BNCOMPLETE  80$             ; br = no error
                         24 50    E9  1CB0                       BLBC    R0, 80$
                                     1CB3  9523         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_INT_ERR,PRLINK=DATA_ERR_CONT
                   00001E4D'EF    DF  1CB3                       PUSHAL  DATA_ERR_CONT
                   000019D1'EF    DF  1CB9                       PUSHAL  T_WR_INT_ERR
                            00    DD  1CBF                       PUSHL   #0
                            03    DD  1CC1                       PUSHL   #$ER
                                     1CC3       ;;; TEST 12, SUBTEST 12, ERROR 3
              000100D0 9F   04   FB  1CC3                       CALLS   #$$M, @#DS$ERRHARD
                                     1CCA  9524         $DS_BNCOMPLETE  70$
                         03 50    E9  1CCA                       BLBC    R0, 70$
                                     1CCD  9525         $DS_ABORT       TEST
                            50    D4  1CCD                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                  04  1CCF                       RET                     ; TERMINATE TEST
                                     1CD0  9526 70$:    $DS_ABORT
              00010020 9F   6C   FA  1CD0                       CALLG   (AP), @#DS$ABORT
                                     1CD7  9527                                         ; send data packet
                   00000000'EF    94  1CD7  9528 80$:    CLRB    B_FLAGS                 ; clear byte flags
                   00000001'EF    94  1CDD  9529         CLRB    B_FLAGS2                ; clear byte flags
                   00000002'EF    94  1CE3  9530         CLRB    B_FLAGS3                ; clear byte flags
              55    0000070A'EF    9E  1CE9  9531         MOVAB   B_DATA_PKT_2,R5         ; point to data packet
         57   55    00000084 8F    C1  1CF0  9532         ADDL3   #132,R5,R7              ; load pointer to end of data
```

```
           19   FF7E4CE0 8F    DA  1CF8  9533          MTPR     #DELAY_TIME7,#PR750$_NICR        ; load interval register    5.2
           00000000'8F    10    DA  1CFF  9534          MTPR     #M_TRANSFER,#PR$_ICCS   ; transfer time to ICS
                   19     01    DA  1D06  9535          MTPR     #1,#PR750$_NICR         ; load very long next interval     5.2
    00000000'8F    80000041 8F  DA  1D09  9536          MTPR     #<M_ERROR+M_ENABLE+M_RUN>,#PR$_ICCS ; start timer
           1E      00000040 8F  DA  1D14  9537          MTPR     #M_ENABLE,#PR750$_CSTS  ; enable send interrupt            5.2
                                    1D1B  9538
                                    1D1B  9539 85$:     $DS_BREAK
           00010058 9F    6E    FA  1D1B                         CALLG   (SP), @#DS$BREAK
        08 00000002'EF    01    E0  1D22  9540          BBS      #V_SDONE_FLAG,B_FLAGS3,87$ ; br = finished sending
        E9 00000000'EF    00    E1  1D2A  9541          BBC      #V_CLOCK_FLAG,B_FLAGS,85$ ; br = no timeout
                   1E     00    DA  1D32  9542 87$:     MTPR     #0,#PR750$_CSTS         ; disable send interrupt           5.2
                   57     55    D1  1D35  9543          CMPL     R5,R7                   ; check number of bytes sent
                          24    13  1D38  9544          BEQL     100$                    ; br = correct number of bytes sent
                                    1D3A  9545          $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_WR_INT_ERR,PRLINK=SEND_INT_ERR
              00001AC9'EF      DF   1D3A                         PUSHAL  SEND_INT_ERR
              000019D1'EF      DF   1D40                         PUSHAL  T_WR_INT_ERR
                          00   DD   1D46                         PUSHL   #0
                          04   DD   1D48                         PUSHL   #$ER
                                    1D4A          ;;; TEST 12, SUBTEST 12, ERROR 4
           000100D0 9F    04    FB  1D4A                         CALLS   #$$M, @#DS$ERRHARD
                                    1D51  9546          $DS_BNCOMPLETE  90$
                      03 50     E9  1D51                         BLBC    R0, 90$
                                    1D54  9547          $DS_ABORT       TEST
                          50    D4  1D54                         CLRL    R0                      ; SEND A WARNING TO THE SUPERVISOR
                          04    1D56                             RET                             ; TERMINATE TEST
                                    1D57  9548 90$:     $DS_ABORT
           00010020 9F    6C    FA  1D57                         CALLG   (AP), @#DS$ABORT
                                    1D5E  9549                                                   ; read end packet
              00000002'EF    96     1D5E  9550 100$:    INCB     B_FLAGS3                ; set data done flag
           54    FF7E4CE0 8F    D0  1D64  9551          MOVL     #DELAY_TIME7,R4         ; load receive packet delay time
           55    00000000'FF    9E  1D6B  9552          MOVAB    @A_BUFFER_BEGIN,R5      ; load address to store actual data
               57   55    0E    C1  1D72  9553          ADDL3    #14,R5,R7               ; load last expected address of data
              000020C8'EF    16     1D76  9554          JSB      MRECEIVE_TU58           ; go get data from TU58
           5A    00000698'EF    9E  1D7C  9555          MOVAB    MB_END_PACKET_5,R10     ; load address of expected end packet
           5B    00000000'FF    9E  1D83  9556          MOVAB    @A_BUFFER_BEGIN,R11     ; load address of actual end packet
                   57     55    D1  1D8A  9557          CMPL     R5,R7                   ; check for correct number of bytes
                          31    13  1D8D  9558          BEQL     140$                    ; br = correct number of bytes
           00000000'EF    08    88  1D8F  9559          BISB2    #M_END_FLAG,B_FLAGS     ; set end error flag
                                    1D96  9560 105$:    $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_INT_ERR,PRLINK=REC_ERR,-
                                    1D96  9561                           P1=#T_REC_END
              00001D2E'8F      DD   1D96                         PUSHL   #T_REC_END
              00001EB4'EF      DF   1D9C                         PUSHAL  REC_ERR
              000019D1'EF      DF   1DA2                         PUSHAL  T_WR_INT_ERR
                          00   DD   1DA8                         PUSHL   #0
                          05   DD   1DAA                         PUSHL   #$ER
                                    1DAC          ;;; TEST 12, SUBTEST 12, ERROR 5
           000100D0 9F    05    FB  1DAC                         CALLS   #$$M, @#DS$ERRHARD
                                    1DB3  9562          $DS_BNCOMPLETE  110$
                      03 50     E9  1DB3                         BLBC    R0, 110$
                                    1DB6  9563          $DS_ABORT       TEST
                          50    D4  1DB6                         CLRL    R0                      ; SEND A WARNING TO THE SUPERVISOR
                          04    1DB8                             RET                             ; TERMINATE TEST
                                    1DB9  9564
                                    1DB9  9565 110$:    $DS_ABORT
           00010020 9F    6C    FA  1DB9                         CALLG   (AP), @#DS$ABORT
```

```
                                 1DC0  9566                                          ; check end packet
                  5C    0E    D0  1DC0  9567 140$:   MOVL    #14,R12                ; load number of bytes to check
                                 1DC3  9568          $DS_BREAK
          00010058 9F    6E    FA  1DC3                        CALLG   (SP), @#DS$BREAK
              0000213B'EF    16    1DCA  9569          JSB     CHECK_PACKET              ; go check end packet
                                 1DD0  9570          $DS_BNCOMPLETE  160$              ; br = no error
                  24 50    E9  1DD0                        BLBC    R0, 160$
                                 1DD3  9571          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_WR_INT_ERR,PRLINK=DATA_ERR_END
              00001E9D'EF    DF  1DD3                        PUSHAL  DATA_ERR_END
              000019D1'EF    DF  1DD9                        PUSHAL  T_WR_INT_ERR
                      00    DD  1DDF                        PUSHL   #0
                      06    DD  1DE1                        PUSHL   #$ER
                                 1DE3          ;;; TEST 12, SUBTEST 12, ERROR 6
          000100D0 9F    04    FB  1DE3                        CALLS   #$$M, @#DS$ERRHARD
                                 1DEA  9572          $DS_BNCOMPLETE  150$
                  03 50    E9  1DEA                        BLBC    R0, 150$
                                 1DED  9573          $DS_ABORT       TEST
                      50    D4  1DED                        CLRL    R0                ; SEND A WARNING TO THE SUPERVISOR
                            04  1DEF                        RET                       ; TERMINATE TEST
                                 1DF0  9574 150$:   $DS_ABORT
          00010020 9F    6C    FA  1DF0                        CALLG   (AP), @#DS$ABORT
                                 1DF7  9575 160$:
                                 1DF7  9576          $DS_ENDSUB
                                 1DF7          T12_S12_X:
  00010038 9F   00000174'EF    FA  1DF7                        CALLG   $S$, @#DS$ENDSUB
                                 1E02  9577          $DS_PAGE
```

```
1E02                .SBTTL  SUBTEST 12.13:   READ RECORD INTERRUPT
1E02      ;+
1E02      ; SUBTEST DESCRIPTION:
1E02      ;
1E02      ;         This subtest sends the TU58 a command packet to read the last 128
1E02      ;         bytes of data that were written in the previous subtest in the inter-
1E02      ;         rupt-driven mode. A timer is set which should allow sufficient time
1E02      ;         to read a complete record and end packet. An error is reported if
1E02      ;         this does not occur within the allotted time. The end packet and data
1E02      ;         packet are checked for errors. If any errors are encountered they are
1E02      ;         reported.
1E02      ;
1E02      ;
1E02      ; SUBTEST STEPS:
1E02      ;
1E02      ;         SUBTEST READ RECORD INTERRUPT
1E02      ;         :  Clear buffer
1E02      ;         :  Point to read command packet
1E02      ;         :  REPEAT
1E02      ;         :  :  Send command packet byte at a time
1E02      ;         :  :  Set timer
1E02      ;         :  :  IF timer expired and no response
1E02      ;         :  :  :  THEN
1E02      ;         :  :  :  Report no response error
1E02      ;         :  :  :  Exit test
1E02      ;         :  :  ENDIF
1E02      ;         :  : UNTIL all bytes sent
1E02      ;         :  ENDREPEAT
1E02      ;         :  ENABLE read interrupt
1E02      ;         :  SET timer
1E02      ;         :  REPEAT
1E02      ;         :  :  IF interrupt occurs
1E02      ;         :  :  :  THEN
1E02      ;         :  :  :  :  Read byte
1E02      ;         :  :  ENDIF
1E02      ;         :  : UNTIL timer expired
1E02      ;         :  ENDREPEAT
1E02      ;         :  IF expected byte count NEQ actual byte count
1E02      ;         :  :  THEN
1E02      ;         :  :  :  DISABLE read interrupt
1E02      ;         :  :  :  Report error
1E02      ;         :  :  :  Exit test
1E02      ;         :  ENDIF
1E02      ;         :  DISABLE read interrupt
1E02      ;         :  Point to expected data packet
1E02      ;         :  REPEAT
1E02      ;         :  :  IF data expected NEQ data received
1E02      ;         :  :  :  THEN
1E02      ;         :  :  :  Report data error
1E02      ;         :  :  :  Exit subtest
1E02      ;         :  :  ENDIF
1E02      ;         :  :  Point to next byte of data
1E02      ;         :  : UNTIL all data in packet checked
1E02      ;         :  ENDREPEAT
1E02      ;         :  Point to expected end packet
```

```
                                  1E02  9634 ;          :  REPEAT
                                  1E02  9635 ;          :  :  IF expected end packet data NEQ received end packet data
                                  1E02  9636 ;          :  :  :  THEN
                                  1E02  9637 ;          :  :  :  Report end packet data error
                                  1E02  9638 ;          :  :  :  Exit subtest
                                  1E02  9639 ;          :  :  ENDIF
                                  1E02  9640 ;          :  : UNTIL all end packet bytes checked
                                  1E02  9641 ;          :  ENDREPEAT
                                  1E02  9642 ;          ENDSUBTEST READ RECORD INTERRUPT
                                  1E02  9643 ;
                                  1E02  9644 ;
                                  1E02  9645 ;
                                  1E02  9646 ;.page
                                  1E02  9647 ;-
                                  1E02  9648            $DS_BGNSUB
                                  1E02       T12_S13::
     00010030 9F   00000180'EF  FA  1E02                       CALLG    $S$, @#DS$BGNSUB
                                  1E0D  9649            $DS_SETVEC_S     VECTOR=#SCB$L_CSRD,SRVADR=MREAD_SERVICE
                           00   DD  1E0D                       PUSHL    #0
                   0000216C'EF  DF  1E0F                       PUSHAL   MREAD_SERVICE
                   000000F0 8F  DD  1E15                       PUSHL    #SCB$L_CSRD
            00010160 9F    03   FB  1E1B                       CALLS    #3, @#DS$SETVEC
                   00000000'EF  94  1E22  9650            CLRB     B_FLAGS                    ; clear byte flags
                   00000001'EF  94  1E28  9651            CLRB     B_FLAGS2                   ; clear byte flags
            58     00000000'FF  9E  1E2E  9652            MOVAB    @A_BUFFER_BEGIN,R8         ; load buffer start address
            59     00000400 8F  D0  1E35  9653            MOVL     #1024,R9                   ; load number of bytes to clear
                   00002134'EF  16  1E3C  9654            JSB      CLEAR_BUFFER               ; go clear buffer
            1C     00000040 8F  DA  1E42  9655            MTPR     #M_ENABLE,#PR750$_CSRS     ; enable read interrupt          5.2
            53     00000000'FF  9E  1E49  9656            MOVAB    @A_BUFFER_BEGIN,R3         ; load buffer address
       59   00000092 8F    53   C1  1E50  9657            ADDL3    R3,#146,R9                 ; load expected end of buffer
            55     0000036C'EF  9E  1E58  9658            MOVAB    MB_READ_3_CMD,R5                   ; point to read record command pac -
cont> ket
                     57    0E   D0  1E5F  9659            MOVL     #14,R7                     ; load byte count
                   0000203D'EF  16  1E62  9660            JSB      SEND_TU58                  ; send command packet
         24 00000000'EF    00   E1  1E68  9661            BBC      #V_CLOCK_FLAG,B_FLAGS,20$ ; br = no timeout error
                                  1E70  9662            $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_INT_ERR,PRLINK=NO_READY
                   00001BEB'EF  DF  1E70                       PUSHAL   NO_READY
                   0000199F'EF  DF  1E76                       PUSHAL   T_RD_INT_ERR
                           00   DD  1E7C                       PUSHL    #0
                           01   DD  1E7E                       PUSHL    #$ER
                                  1E80       ;;; TEST 12, SUBTEST 13, ERROR 1
            000100D0 9F    04   FB  1E80                       CALLS    #$$M, @#DS$ERRHARD
                                  1E87  9663            $DS_BNCOMPLETE   10$
                        03 50   E9  1E87                       BLBC     R0, 10$
                                  1E8A  9664            $DS_ABORT        TEST
                           50   D4  1E8A                       CLRL     R0                         ; SEND A WARNING TO THE SUPERVISOR
                           04       1E8C                       RET                                 ; TERMINATE TEST
                                  1E8D  9665 10$:       $DS_ABORT
            00010020 9F    6C   FA  1E8D                       CALLG    (AP), @#DS$ABORT
                   00000000'EF  94  1E94  9666 20$:       CLRB     B_FLAGS                    ; clear byte flags
            19     FF7E4CE0 8F  DA  1E9A  9667            MTPR     #DELAY_TIME7,#PR750$_NICR         ; load timer interval    5.2
            00000000'8F    10   DA  1EA1  9668            MTPR     #M_TRANSFER,#PR$_ICCS      ; transfer time to ICS
                     19    01   DA  1EA8  9669            MTPR     #1,#PR750$_NICR            ; load very long next interval  5.2
     00000000'8F   80000041 8F  DA  1EAB  9670            MTPR     #<M_ERROR+M_ENABLE+M_RUN>,#PR$_ICCS ; start timer
                                  1EB6  9671 30$:       $DS_BREAK
            00010058 9F    6E   FA  1EB6                       CALLG    (SP), @#DS$BREAK
       F1 00000000'EF   00   E1  1EBD  9672          BBC     #V_CLOCK_FLAG,B_FLAGS,30$ ; br = no timeout
                 1C     00   DA  1EC5  9673          MTPR    #0,#PR750$_CSRS         ; disable read interrupt        5.2
          5B   00000000'FF   9E  1EC8  9674          MOVAB   @A_BUFFER_BEGIN,R11     ; load address of actual data
          5A   0000070A'EF   9E  1ECF  9675          MOVAB   B_DATA_PKT_2,R10        ; load address of expected data
                 59     53   D1  1ED6  9676          CMPL    R3,R9                   ; correct number of bytes received ?
                        34   13  1ED9  9677          BEQL    50$                     ; br = correct number of bytes received
          00000000'EF   04   88  1EDB  9678          BISB2   #M_DATA_FLAG,B_FLAGS    ; set data error flag
                 55     53   D0  1EE2  9679          MOVL    R3,R5                   ; point to last address loaded
                                 1EE5  9680          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_INT_ERR,PRLINK=REC_ERR,-
                                 1EE5  9681                           P1=#T_REC_DATA
               00001D68'8F   DD  1EE5                         PUSHL   #T_REC_DATA
               00001EB4'EF   DF  1EEB                         PUSHAL  REC_ERR
               0000199F'EF   DF  1EF1                         PUSHAL  T_RD_INT_ERR
                        00   DD  1EF7                         PUSHL   #0
                        02   DD  1EF9                         PUSHL   #$ER
                                 1EFB        ;;; TEST 12, SUBTEST 13, ERROR 2
          000100D0 9F   05   FB  1EFB                         CALLS   #$$M, @#DS$ERRHARD
                                 1F02  9682          $DS_BNCOMPLETE  40$
                     03 50   E9  1F02                         BLBC    R0, 40$
                                 1F05  9683          $DS_ABORT       TEST
                        50   D4  1F05                         CLRL    R0                 ; SEND A WARNING TO THE SUPERVISOR
                        04   04  1F07                         RET                        ; TERMINATE TEST
                                 1F08  9684 40$:     $DS_ABORT
          00010020 9F   6C   FA  1F08                         CALLG   (AP), @#DS$ABORT
          5C   00000084 8F   D0  1F0F  9685 50$:     MOVL    #132,R12                    ; load number of bytes to check
                                 1F16  9686          $DS_BREAK
          00010058 9F   6E   FA  1F16                         CALLG   (SP), @#DS$BREAK
               0000213B'EF   16  1F1D  9687          JSB     CHECK_PACKET                ; go check data packet
                                 1F23  9688          $DS_BNCOMPLETE  70$                 ; br = no data error
                     24 50   E9  1F23                         BLBC    R0, 70$
                                 1F26  9689          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_INT_ERR,PRLINK=DATA_ERR_DATA
               00001E75'EF   DF  1F26                         PUSHAL  DATA_ERR_DATA
               0000199F'EF   DF  1F2C                         PUSHAL  T_RD_INT_ERR
                        00   DD  1F32                         PUSHL   #0
                        03   DD  1F34                         PUSHL   #$ER
                                 1F36        ;;; TEST 12, SUBTEST 13, ERROR 3
          000100D0 9F   04   FB  1F36                         CALLS   #$$M, @#DS$ERRHARD
                                 1F3D  9690          $DS_BNCOMPLETE  60$
                     03 50   E9  1F3D                         BLBC    R0, 60$
                                 1F40  9691          $DS_ABORT       TEST
                        50   D4  1F40                         CLRL    R0                 ; SEND A WARNING TO THE SUPERVISOR
                        04       1F42                         RET                        ; TERMINATE TEST
                                 1F43  9692 60$:     $DS_ABORT
          00010020 9F   6C   FA  1F43                         CALLG   (AP), @#DS$ABORT
          5A   00000698'EF   9E  1F4A  9693 70$:     MOVAB   MB_END_PACKET_5,R10     ; load address of expected end packet
          5B   00000084 8F   C0  1F51  9694          ADDL2   #132,R11                ; adjust pointer to actual end packet
                 5C     0E   D0  1F58  9695          MOVL    #14,R12                 ; load number of bytes to check
                                 1F5B  9696          $DS_BREAK
          00010058 9F   6E   FA  1F5B                         CALLG   (SP), @#DS$BREAK
               0000213B'EF   16  1F62  9697          JSB     CHECK_PACKET            ; go check end packet
                                 1F68  9698          $DS_BNCOMPLETE  90$             ; br = no data error
                     24 50   E9  1F68                         BLBC    R0, 90$
                                 1F6B  9699          $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_RD_INT_ERR,PRLINK=DATA_ERR_END
               00001E9D'EF   DF  1F6B                         PUSHAL  DATA_ERR_END
               0000199F'EF   DF  1F71                         PUSHAL  T_RD_INT_ERR
```

```
                          00    DD  1F77                           PUSHL    #0
                          04    DD  1F79                           PUSHL    #$ER
                                    1F7B        ;;; TEST 12, SUBTEST 13, ERROR 4
              000100D0 9F 04    FB  1F7B                           CALLS    #$$M, a#DS$ERRHARD
                                    1F82  9700          $DS_BNCOMPLETE  80$
                       03 50    E9  1F82                           BLBC     R0, 80$
                                    1F85  9701          $DS_ABORT       TEST
                          50    D4  1F85                           CLRL     R0                   ; SEND A WARNING TO THE SUPERVISOR
                                04  1F87                           RET                           ; TERMINATE TEST
                                    1F88  9702 80$:     $DS_ABORT
              00010020 9F 6C    FA  1F88                           CALLG    (AP), a#DS$ABORT
                                    1F8F  9703 90$:
                                    1F8F  9704          $DS_ENDSUB
                                    1F8F        T12_S13_X:
      00010038 9F   00000180'EF FA  1F8F                           CALLG    $S$, a#DS$ENDSUB
                                    1F9A  9705          $DS_PAGE
```

```
1F9A          .SBTTL  SUBTEST 12.14:  IPL CHECK
1F9A  ;+
1F9A  ; SUBTEST DESCRIPTION:
1F9A  ;
1F9A  ;       This subtest checks that the TU58 interrupt occurs at the cor-
1F9A  ;       rect interrupt priority level.  The IPL is set one level above
1F9A  ;       that at which a TU58 interrupt should occur.  A NOP command is
1F9A  ;       issued to cause the return  of an end packet in interrupt mode.
1F9A  ;       If an interrupt occurs an error is reported.  The IPL is then
1F9A  ;       decremented and if an interrupt does not occur within a reason-
1F9A  ;       interval an error is reported.  There are no data checks made.
1F9A  ;
1F9A  ;
1F9A  ; SUBTEST STEPS:
1F9A  ;
1F9A  ;       SUBTEST IPL CHECK
1F9A  ;       : Point to NOP command packet
1F9A  ;       : REPEAT
1F9A  ;       : : SEND command packet byte at a time
1F9A  ;       : : SET timer
1F9A  ;       : : IF timer expired AND no response
1F9A  ;       : : : THEN
1F9A  ;       : : : : REPORT no response error
1F9A  ;       : : : : Exit test
1F9A  ;       : : ENDIF
1F9A  ;       : : UNTIL all bytes sent
1F9A  ;       : ENDREPEAT
1F9A  ;       : SET IPL above read interrupt
1F9A  ;       : SET timer
1F9A  ;       : ENABLE read interrupt
1F9A  ;       : IF interrupt occurs before timeout
1F9A  ;       : : THEN
1F9A  ;       : : : DISABLE read interrupt
1F9A  ;       : : : REPORT error
1F9A  ;       : : : Abort test
1F9A  ;       : : ENDIF
1F9A  ;       : DISABLE read interrupt
1F9A  ;       : DECR IPL
1F9A  ;       : SET timer
1F9A  ;       : ENABLE read interrupt
1F9A  ;       : IF timer expires AND no interrupt
1F9A  ;       : : THEN
1F9A  ;       : : : DISABLE read interrupt
1F9A  ;       : : : REPORT error
1F9A  ;       : : : Abort test
1F9A  ;       : ENDIF
1F9A  ;       : DISABLE read interrupt
1F9A  ;       : CLEAR IPL
1F9A  ;       ENDSUBTEST IPL CHECK
1F9A  ;
1F9A  ;
1F9A  ;
1F9A  ;.page
1F9A  ;-
1F9A
```

```
                                     1F9A  9762         $DS_BGNSUB
                                     1F9A       T12_S14::
     00010030 9F   0000018C'EF   FA  1F9A                       CALLG   $S$, a#DS$BGNSUB
                                     1FA5  9763         $DS_CLRVEC_S    VECTOR=#SCB$L_CSRD
                   000000F0 8F   DD  1FA5                       PUSHL   #SCB$L_CSRD
              00010168 9F   01   FB  1FAB                       CALLS   #1, a#DS$CLRVEC
                                     1FB2  9764         $DS_SETVEC_S    VECTOR=#SCB$L_CSRD,SRVADR=MREAD_SERVICE2
                            00   DD  1FB2                       PUSHL   #0
                   000021AC'EF   DF  1FB4                       PUSHAL  MREAD_SERVICE2
                   000000F0 8F   DD  1FBA                       PUSHL   #SCB$L_CSRD
              00010160 9F   03   FB  1FC0                       CALLS   #3, a#DS$SETVEC
                   00000000'EF   94  1FC7  9765         CLRB    B_FLAGS                 ; clear byte flags
                   00000001'EF   94  1FCD  9766         CLRB    B_FLAGS2                ; clear byte flags
                   00000002'EF   94  1FD3  9767         CLRB    B_FLAGS3                ; clear byte flags
              54   FFFB6C20 8F   D0  1FD9  9768         MOVL    #DELAY_TIME1,R4         ; load send packet delay time
              55   0000030A'EF   9E  1FE0  9769         MOVAB   MB_NOP_CMD,R5           ; point to init command packet
                       57   0E   D0  1FE7  9770         MOVL    #14,R7                  ; load transfer byte count
                   0000203D'EF   16  1FEA  9771         JSB     SEND_TU58               ; send command packet
           24 00000000'EF   00   E1  1FF0  9772         BBC     #V_CLOCK_FLAG,B_FLAGS,20$ ; br = no timeout error
                                     1FF8  9773         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_IPL_ERR,PRLINK=NO_READY
                   00001BEB'EF   DF  1FF8                       PUSHAL  NO_READY
                   00001A04'EF   DF  1FFE                       PUSHAL  T_IPL_ERR
                            00   DD  2004                       PUSHL   #0
                            01   DD  2006                       PUSHL   #$ER
                                     2008       ;;; TEST 12, SUBTEST 14, ERROR 1
              000100D0 9F   04   FB  2008                       CALLS   #$$M, a#DS$ERRHARD
                                     200F  9774         $DS_BNCOMPLETE  10$
                         03 50   E9  200F                       BLBC    R0, 10$
                                     2012  9775         $DS_ABORT       TEST
                            50   D4  2012                       CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  2014                       RET                     ; TERMINATE TEST
                                     2015  9776 10$:    $DS_ABORT
              00010020 9F   6C   FA  2015                       CALLG   (AP), a#DS$ABORT
                                     201C  9777
                   00000000'EF   94  201C  9778 20$:    CLRB    B_FLAGS                 ; clear byte flags
                       56   1D   DB  2022  9779         MFPR    #PR750$_CSRD,R6         ; dump buffer to clear done     5.2
                                     2025  9780         $DS_SETIPL_S    LEVEL=#^X17     ; set IPL above read interrupt
                            17   DD  2025                       PUSHL   #^X17
              00010178 9F   01   FB  2027                       CALLS   #1, a#DS$SETIPL
              19   FFF3CB00 8F   DA  202E  9781         MTPR    #DELAY_TIME3,#PR750$_NICR    ; load timer interval      5.2
              00000000'8F   10   DA  2035  9782         MTPR    #M_TRANSFER,#PR$_ICCS   ; transfer time to ICS
                       19   01   DA  203C  9783         MTPR    #1,#PR750$_NICR         ; load very long next interval  5.2
     00000000'8F   80000041 8F   DA  203F  9784         MTPR    #<M_ERROR+M_ENABLE+M_RUN>,#PR$_ICCS ; start timer
              1C   00000040 8F   DA  204A  9785         MTPR    #M_ENABLE,#PR750$_CSRS  ; enable read interrupt         5.2
                                     2051  9786 30$:    $DS_BREAK                       ; enable break
              00010058 9F   6E   FA  2051                       CALLG   (SP), a#DS$BREAK
           F1 00000000'EF   00   E1  2058  9787         BBC     #V_CLOCK_FLAG,B_FLAGS,30$ ; br = no timeout
           27 00000002'EF   02   E1  2060  9788         BBC     #V_RD_INT_FLAG,B_FLAGS3,50$ ; br = no interrupt
                       1C   00   DA  2068  9789         MTPR    #0,#PR750$_CSRS         ; disable read interrupt        5.2
                                     206B  9790         $DS_ERRHARD_S   ,UNIT=#0,MSGADR=T_IPL_ERR,PRLINK=INT_ERR
                   00001F3D'EF   DF  206B                       PUSHAL  INT_ERR
                   00001A04'EF   DF  2071                       PUSHAL  T_IPL_ERR
                            00   DD  2077                       PUSHL   #0
                            02   DD  2079                       PUSHL   #$ER
                                     207B       ;;; TEST 12, SUBTEST 14, ERROR 2
           000100D0 9F    04   FB  207B                      CALLS   #$$M, @#DS$ERRHARD
                                   2082  9791        $DS_BNCOMPLETE  40$
                        03 50  E9  2082                      BLBC    R0, 40$
                                   2085  9792        $DS_ABORT       TEST
                           50  D4  2085                      CLRL    R0                 ; SEND A WARNING TO THE SUPERVISOR
                               04  2087                      RET                        ; TERMINATE TEST
                                   2088  9793 40$:   $DS_ABORT
           00010020 9F    6C   FA  2088                      CALLG   (AP), @#DS$ABORT
                                   208F  9794
           53    00000000'FF   9E  208F  9795 50$:   MOVAB   @A_BUFFER_BEGIN,R3       ; load buffer address
                      1C    00  DA  2096  9796        MTPR    #0,#PR750$_CSRS          ; disable read interrupt        5.2
                 00000002'EF   94  2099  9797        CLRB    B_FLAGS3                 ; clear byte flags
           19    FF7E4CE0 8F   DA  209F  9798        MTPR    #DELAY_TIME7,#PR750$_NICR        ; load timer interval   5.2
           00000000'8F    10   DA  20A6  9799        MTPR    #M_TRANSFER,#PR$_ICCS    ; transfer time to ICS
                      19    01  DA  20AD  9800        MTPR    #1,#PR750$_NICR          ; load very long next interval  5.2
   00000000'8F    80000041 8F  DA  20B0  9801        MTPR    #<M_ERROR+M_ENABLE+M_RUN>,#PR$_ICCS ; start timer
           1C    00000040 8F   DA  20BB  9802        MTPR    #M_ENABLE,#PR750$_CSRS   ; enable read interrupt         5.2
                                   20C2  9803        $DS_SETIPL_S    LEVEL=#^X16      ; set IPL to allow read interrupt
                           16  DD  20C2                      PUSHL   #^X16
           00010178 9F    01   FB  20C4                      CALLS   #1, @#DS$SETIPL
                                   20CB  9804 60$:   $DS_BREAK                          ; enable break
           00010058 9F    6E   FA  20CB                      CALLG   (SP), @#DS$BREAK
        2F 00000002'EF    02   E0  20D2  9805        BBS     #V_RD_INT_FLAG,B_FLAGS3,80$ ; br = interrupt
        E9 00000000'EF    00   E1  20DA  9806        BBC     #V_CLOCK_FLAG,B_FLAGS,60$ ; br = no timeout
                      1C    00  DA  20E2  9807        MTPR    #0,#PR750$_CSRS          ; disable read interrupt        5.2
                                   20E5  9808        $DS_ERRHARD_S    ,UNIT=#0,MSGADR=T_IPL_ERR,PRLINK=NO_INT_ERR
                 00001F4B'EF   DF  20E5                      PUSHAL  NO_INT_ERR
                 00001A04'EF   DF  20EB                      PUSHAL  T_IPL_ERR
                           00  DD  20F1                      PUSHL   #0
                           03  DD  20F3                      PUSHL   #$ER
                                   20F5       ;;; TEST 12, SUBTEST 14, ERROR 3
           000100D0 9F    04   FB  20F5                      CALLS   #$$M, @#DS$ERRHARD
                                   20FC  9809        $DS_BNCOMPLETE  70$
                        03 50  E9  20FC                      BLBC    R0, 70$
                                   20FF  9810        $DS_ABORT       TEST
                           50  D4  20FF                      CLRL    R0                 ; SEND A WARNING TO THE SUPERVISOR
                               04  2101                      RET                        ; TERMINATE TEST
                                   2102  9811 70$:   $DS_ABORT
           00010020 9F    6C   FA  2102                      CALLG   (AP), @#DS$ABORT
                                   2109  9812
                      1C    00  DA  2109  9813 80$:   MTPR    #0,#PR750$_CSRS          ; disable interrupt             5.2
                                   210C  9814        $DS_SETIPL_S    LEVEL=#0         ; restore IPL
                           00  DD  210C                      PUSHL   #0
           00010178 9F    01   FB  210E                      CALLS   #1, @#DS$SETIPL
                                   2115  9815
                                   2115  9816        $DS_ENDSUB
                                   2115       T12_S14_X:
   00010038 9F    0000018C'EF  FA  2115                      CALLG   $S$, @#DS$ENDSUB
                                   2120  9817
                                   2120  9818        $DS_CLRVEC_S    VECTOR=#SCB$L_TIMER
                 000000C0 8F   DD  2120                      PUSHL   #SCB$L_TIMER
           00010168 9F    01   FB  2126                      CALLS   #1, @#DS$CLRVEC
                                   212D  9819        $DS_CLRVEC_S    VECTOR=#SCB$L_CSRD
                 000000F0 8F   DD  212D                      PUSHL   #SCB$L_CSRD
           00010168 9F    01   FB  2133                      CALLS   #1, @#DS$CLRVEC
```

```
                                 213A  9820        $DS_CLRVEC_S    VECTOR=#SCB$L_CSTD
            000000F4 8F     DD   213A                      PUSHL   #SCB$L_CSTD
       00010168 9F    01    FB   2140                      CALLS   #1, @#DS$CLRVEC
                                 2147  9821
                                 2147  9822        $DS_ENDTEST
                50    01    D0   2147                      MOVL    #1, R0          ; NORMAL EXIT
                                 214A          TEST_012_X::
       00010058 9F    6E    FA   214A                      CALLG   (SP), @#DS$BREAK
                            04   2151                      RET                     ; RETURN TO TEST SEQUENCER
                                 2152  9823        $DS_ENDMOD
                            00000000                       .PSECT  $TSTCNT, NOEXE, NOWRT, OVR, LONG
                  0000000C  0000                           .LONG   $TN
                                 0004  9824
                                 0004  9825        .END
```

```
$$$                           = 0000018C R     05        B_DATA_PKT_12                   00000BA7 R     01
$$ARGS                        = 0000000A                 B_DATA_PKT_13                   00000C2B R     01
$$E                           = 00000004                 B_DATA_PKT_14                   00000CAF R     01
$$M                           = 00000004                 B_DATA_PKT_15                   00000D33 R     01
$$N                           = 00000000                 B_DATA_PKT_16                   00000DB7 R     01
$$S                           = FFFFFFFF                 B_DATA_PKT_17                   00000E3B R     01
$$T1                          = 0000002C                 B_DATA_PKT_18                   00000EBF R     01
$ENV                          = 00000001  G              B_DATA_PKT_19                   00000F43 R     01
$ER                           = FFFFFFFF                 B_DATA_PKT_2                    0000070A R     01
$MO                           = 00000001  G              B_DATA_PKT_20                   00000FC7 R     01
$S$                           = 0000018C R     05        B_DATA_PKT_21                   0000104B R     01
$ST                           = 00000000                 B_DATA_PKT_22                   000010B3 R     01
$TN                           = 0000000C                 B_DATA_PKT_23                   00001137 R     01
ALL                           = 00000009  G              B_DATA_PKT_24                   000011BB R     01
A_BUFFER_BEGIN                  ********   X   01        B_DATA_PKT_25                   0000123F R     01
BASE_009                        00000000 R     03        B_DATA_PKT_26                   000012A7 R     01
BASE_010                        00000000 R     06        B_DATA_PKT_3                    0000078E R     01
BASE_011                        00000000 R     07        B_DATA_PKT_4                    00000812 R     01
BASE_012                        00000000 R     08        B_DATA_PKT_5                    00000817 R     01
BIT...                        = 00000005                 B_DATA_PKT_6                    0000089B R     01
BLOCK_00                      = 00000000                 B_DATA_PKT_7                    0000091F R     01
BLOCK_01                      = 00000001                 B_DATA_PKT_8                    000009A3 R     01
BLOCK_02                      = 00000002                 B_DATA_PKT_9                    00000A27 R     01
BLOCK_04                      = 00000004                 B_DIAGNOSE_CMD                  00000040 R     01
BLOCK_05                      = 00000005                 B_END_PACKET_1                  000005D4 R     01
BLOCK_06                      = 00000006                 B_END_PACKET_10                 00000652 R     01
BLOCK_0A                      = 0000000A                 B_END_PACKET_2                  000005E2 R     01
BLOCK_0C                      = 0000000C                 B_END_PACKET_3                  000005F0 R     01
BLOCK_0D                      = 0000000D                 B_END_PACKET_4                  000005FE R     01
BLOCK_0E                      = 0000000E                 B_END_PACKET_5                  0000060C R     01
BLOCK_0F                      = 0000000F                 B_END_PACKET_6                  0000061A R     01
BLOCK_15                      = 00000015                 B_END_PACKET_7                  00000628 R     01
BLOCK_5A                      = 0000005A                 B_END_PACKET_8                  00000636 R     01
BLOCK_80                      = 00000080                 B_END_PACKET_9                  00000644 R     01
BLOCK_8F                      = 0000008F                 B_FLAGS                         00000000 R     01
BUFFER_SIZE                   = 00000600                 B_FLAGS2                        00000001 R     01
BUF_LIM                         00000013 R     01        B_FLAGS3                        00000002 R     01
BYTE_COUNT_00                 = 00000000                 B_INIT_CMD                      00000024 R     01
BYTE_COUNT_01                 = 00000001                 B_INIT_DRIVE                    0000001F R     01
BYTE_COUNT_02                 = 00000002                 B_NOP_CMD                       00000032 R     01
BYTE_COUNT_04                 = 00000004                 B_POS_1_CMD                     000002EE R     01
BYTE_COUNT_0A                 = 0000000A                 B_READ_10_CMD                   00000174 R     01
BYTE_COUNT_12                 = 00000012                 B_READ_11_CMD                   00000190 R     01
BYTE_COUNT_1A                 = 0000001A                 B_READ_12_CMD                   000001AC R     01
BYTE_COUNT_64                 = 00000064                 B_READ_13_CMD                   000001C8 R     01
BYTE_COUNT_74                 = 00000074                 B_READ_14_CMD                   000001E4 R     01
BYTE_COUNT_80                 = 00000080                 B_READ_15_CMD                   00000200 R     01
BYTE_COUNT_F4                 = 000000F4                 B_READ_16_CMD                   0000021C R     01
B_CONTINUE                      00000023 R     01        B_READ_17_CMD                   00000238 R     01
B_COUNT                         00000003 R     01        B_READ_18_CMD                   00000254 R     01
B_COUNT1                        00000004 R     01        B_READ_19_CMD                   00000270 R     01
B_COUNT2                        00000005 R     01        B_READ_1_CMD                    0000004E R     01
B_DATA_PKT_1                    000006EC R     01        B_READ_20_CMD                   0000028C R     01
B_DATA_PKT_10                   00000AAB R     01        B_READ_21_CMD                   000002A8 R     01
B_DATA_PKT_11                   00000B2F R     01        B_READ_22_CMD                   000002C4 R     01
```

```
B_READ_23_CMD                   000002D2 R      01        CHECKSUM_14                   = 00000014
B_READ_24_CMD                   000002E0 R      01        CHECKSUM_15                   = 00000015
B_READ_2_CMD                    00000078 R      01        CHECKSUM_16                   = 00000016
B_READ_3_CMD                    00000094 R      01        CHECKSUM_19                   = 00000019
B_READ_4_CMD                    000000B0 R      01        CHECKSUM_1A                   = 0000001A
B_READ_5_CMD                    000000E8 R      01        CHECKSUM_1C                   = 0000001C
B_READ_6_CMD                    000000F6 R      01        CHECKSUM_1D                   = 0000001D
B_READ_7_CMD                    0000012E R      01        CHECKSUM_2D                   = 0000002D
B_READ_8_CMD                    0000013C R      01        CHECKSUM_36                   = 00000036
B_READ_9_CMD                    00000158 R      01        CHECKSUM_41                   = 00000041
B_RESPONSE                      00000012 R      01        CHECKSUM_42                   = 00000042
B_WRITE_10_CMD                  00000120 R      01        CHECKSUM_43                   = 00000043
B_WRITE_11_CMD                  0000014A R      01        CHECKSUM_54                   = 00000054
B_WRITE_12_CMD                  00000166 R      01        CHECKSUM_5C                   = 0000005C
B_WRITE_13_CMD                  00000182 R      01        CHECKSUM_6E                   = 0000006E
B_WRITE_14_CMD                  0000019E R      01        CHECKSUM_6F                   = 0000006F
B_WRITE_15_CMD                  000001BA R      01        CHECKSUM_72                   = 00000072
B_WRITE_16_CMD                  000001D6 R      01        CHECKSUM_75                   = 00000075
B_WRITE_17_CMD                  000001F2 R      01        CHECKSUM_79                   = 00000079
B_WRITE_18_CMD                  0000020E R      01        CHECKSUM_80                   = 00000080
B_WRITE_19_CMD                  0000022A R      01        CHECKSUM_82                   = 00000082
B_WRITE_1_CMD                   0000005C R      01        CHECKSUM_83                   = 00000083
B_WRITE_20_CMD                  00000246 R      01        CHECKSUM_84                   = 00000084
B_WRITE_21_CMD                  00000262 R      01        CHECKSUM_85                   = 00000085
B_WRITE_22_CMD                  0000027E R      01        CHECKSUM_88                   = 00000088
B_WRITE_23_CMD                  0000029A R      01        CHECKSUM_89                   = 00000089
B_WRITE_24_CMD                  000002B6 R      01        CHECKSUM_8A                   = 0000008A
B_WRITE_2_CMD                   0000006A R      01        CHECKSUM_8B                   = 0000008B
B_WRITE_3_CMD                   00000086 R      01        CHECKSUM_8C                   = 0000008C
B_WRITE_4_CMD                   000000A2 R      01        CHECKSUM_8D                   = 0000008D
B_WRITE_5_CMD                   000000BE R      01        CHECKSUM_90                   = 00000090
B_WRITE_6_CMD                   000000CC R      01        CHECKSUM_91                   = 00000091
B_WRITE_7_CMD                   000000DA R      01        CHECKSUM_92                   = 00000092
B_WRITE_8_CMD                   00000104 R      01        CHECKSUM_93                   = 00000093
B_WRITE_9_CMD                   00000112 R      01        CHECKSUM_94                   = 00000094
B_ZEROS                         00000021 R      01        CHECKSUM_95                   = 00000095
CEP_FUNCTIONAL                = 00000000                  CHECKSUM_98                   = 00000098
CEP_REPAIR                    = 00000001                  CHECKSUM_A6                   = 000000A6
CHECKSUM_01                   = 00000001                  CHECKSUM_A9                   = 000000A9
CHECKSUM_02                   = 00000002                  CHECKSUM_AA                   = 000000AA
CHECKSUM_03                   = 00000003                  CHECKSUM_AB                   = 000000AB
CHECKSUM_04                   = 00000004                  CHECKSUM_AE                   = 000000AE
CHECKSUM_05                   = 00000005                  CHECKSUM_BA                   = 000000BA
CHECKSUM_08                   = 00000008                  CHECKSUM_BF                   = 000000BF
CHECKSUM_09                   = 00000009                  CHECKSUM_C0                   = 000000C0
CHECKSUM_0A                   = 0000000A                  CHECKSUM_C1                   = 000000C1
CHECKSUM_0B                   = 0000000B                  CHECKSUM_C2                   = 000000C2
CHECKSUM_0C                   = 0000000C                  CHECKSUM_C5                   = 000000C5
CHECKSUM_0D                   = 0000000D                  CHECKSUM_C9                   = 000000C9
CHECKSUM_0E                   = 0000000E                  CHECKSUM_DE                   = 000000DE
CHECKSUM_0F                   = 0000000F                  CHECKSUM_DF                   = 000000DF
CHECKSUM_10                   = 00000010                  CHECKSUM_E0                   = 000000E0
CHECKSUM_11                   = 00000011                  CHECKSUM_E5                   = 000000E5
CHECKSUM_12                   = 00000012                  CHECKSUM_E8                   = 000000E8
CHECKSUM_13                   = 00000013                  CHECKSUM_F4                   = 000000F4
```

```
CHECKSUM_F7                     = 000000F7                  DELAY_TIME7                     = FF7E4CE0
CHECKSUM_FA                     = 000000FA                  DELAY_TIME8                     = FAEF00C0
CHECKSUM_FC                     = 000000FC                  DIAGNOSE                        = 00000007
CHECK_PACKET                      0000213B R     01         DS$ABORT                          00010020
CLEAR_BUFFER                      00002134 R     01         DS$ASKADR                         00010090
COMMAND_FLAG                    = 00000002                  DS$ASKDATA                        00010080
COMMAND_TBL_1                     0000132B R     01         DS$ASKLGCL                        00010098
COMMAND_TBL_2                     00001397 R     01         DS$ASKSTR                         000100A0
COMMAND_TBL_3                     000013D3 R     01         DS$ASKVLD                         00010088
COMMAND_TBL_4                     000014E3 R     01         DS$ATTACH                         000101A8
COMMAND_TBL_5                     00001563 R     01         DS$BGNSUB                         00010030
CONTINUE_FLAG                   = 00000010                  DS$BRANCH                         000100A8
DATA_0000                       = 00000000                  DS$BREAK                          00010058
DATA_0001                       = 00000001                  DS$CANWAIT                        00010070
DATA_0004                       = 00000004                  DS$CHANNEL                        00010180
DATA_000F                       = 0000000F                  DS$CKLOOP                         00010040
DATA_0015                       = 00000015                  DS$CLRVEC                         00010168
DATA_0080                       = 00000080                  DS$CNTRLC                         00010078
DATA_009                          00000014 R     03         DS$CVTREG                         000100B0
DATA_00AA                       = 000000AA                  DS$ENDPASS                        00010010
DATA_00FF                       = 000000FF                  DS$ENDSUB                         00010038
DATA_010                          0000001C R     06         DS$ERRDEV                         000100C8
DATA_0100                       = 00000100                  DS$ERRHARD                        000100D0
DATA_0101                       = 00000101                  DS$ERRPREP                        00010108
DATA_011                          00000020 R     07         DS$ERRSOFT                        000100D8
DATA_0115                       = 00000115                  DS$ERRSYS                         000100C0
DATA_012                          00000028 R     08         DS$ESCAPE                         00010050
DATA_0180                       = 00000180                  DS$FREEDBGSYM                     000101B8
DATA_0204                       = 00000204                  DS$GETBUF                         00010120
DATA_0206                       = 00000206                  DS$GETMEM                         00010130
DATA_025A                       = 0000025A                  DS$GETTERM                        000101C8
DATA_0400                       = 00000400                  DS$GPHARD                         00010018
DATA_0604                       = 00000604                  DS$HELP                           000101B0
DATA_0605                       = 00000605                  DS$INITSCB                        00010170
DATA_0606                       = 00000606                  DS$INLOOP                         00010048
DATA_065A                       = 0000065A                  DS$K_ERROR                      = 00000002
DATA_0800                       = 00000800                  DS$K_NORMAL                     = 00000001
DATA_0960                       = 00000960                  DS$K_SEVERE                     = 00000004
DATA_1400                       = 00001400                  DS$K_SUBSYS                     = 00000066
DATA_1D14                       = 00001D14                  DS$K_WARNING                    = 00000000
DATA_4A77                       = 00004A77                  DS$LOAD                           00010198
DATA_774A                       = 0000774A                  DS$LOADPCS                        000101C0
DATA_7D09                       = 00007D09                  DS$MAPDBGBLOCK                    00010118
DATA_ERR_CONT                     00001E4D R     01         DS$MEMSIZE                        000101D8
DATA_ERR_DATA                     00001E75 R     01         DS$MMOFF                          00010158
DATA_ERR_END                      00001E9D R     01         DS$MMON                           00010150
DATA_FFFF                       = 0000FFFF                  DS$MOVPHY                         00010148
DATA_FLAG                       = 00000001                  DS$MOVVRT                         00010140
DEFAULT                         = 00000000  G               DS$PARSE                          000100B8
DELAY_TIME1                     = FFFB6C20                  DS$PRINTB                         000100E0
DELAY_TIME2                     = FFF85EE0                  DS$PRINTF                         000100F0
DELAY_TIME3                     = FFF3CB00                  DS$PRINTS                         000100F8
DELAY_TIME4                     = FFF0BDC0                  DS$PRINTSIG                       00010100
DELAY_TIME5                     = FFE17B80                  DS$PRINTX                         000100E8
DELAY_TIME6                     = FFC2F700                  DS$PROBE                          000101A0
```

```
DS$RELBUF                         00010128                  END_CODE                          = 00000040
DS$RELMEM                         00010138                  END_FLAG                          = 00000002
DS$SERVICE_DISPATCHER             000101D0                  ENV$M_DOMAIN                      = 00000002
DS$SETIPL                         00010178                  ENV$M_LEVEL                       = 00000001
DS$SETMAP                         00010188                  ENV$M_SUPER                       = 000003FC
DS$SETPRIEXV                      00010110                  ENV$S_DOMAIN                      = 00000001
DS$SETVEC                         00010160                  ENV$S_LEVEL                       = 00000001
DS$SHOCHAN                        00010190                  ENV$S_SUPER                       = 00000008
DS$SUMMARY                        00010028                  ENV$V_DOMAIN                      = 00000001
DS$WAITMS                         00010060                  ENV$V_LEVEL                       = 00000000
DS$WAITUS                         00010068                  ENV$V_SUPER                       = 00000002
DS$_ARITH                       = 006600D0                  ENV$_C                            = 00000000
DS$_ASBE                        = 00660118                  ENV$_FUNCTIONAL                   = 00000000
DS$_BADLINK                     = 006600F0                  ENV$_REPAIR                       = 00000001
DS$_BADTYPE                     = 006600E8                  ENV$_SUPER                        = 00000001
DS$_BIIC                        = 00660120                  ENV$_SYSTEM                       = 00000001
DS$_CHME                        = 006600A8                  ERR$_MSGADR                       = 0000000C
DS$_CHMK                        = 006600E0                  ERR$_NARGS                        = 0000000A
DS$_DEVNAME                     = 00660108                  ERR$_NUM                          = 00000004
DS$_ERROR                       = 00660002                  ERR$_P1                           = 00000014
DS$_FHWE                        = 00660068                  ERR$_P2                           = 00000018
DS$_FRAGBUF                     = 00660080                  ERR$_P3                           = 0000001C
DS$_ICBUSY                      = 006600C8                  ERR$_P4                           = 00000020
DS$_ICERR                       = 006600C0                  ERR$_P5                           = 00000024
DS$_IHWE                        = 00660060                  ERR$_P6                           = 00000028
DS$_ILLCHAR                     = 00660018                  ERR$_POINTER                      = 00000010
DS$_ILLPAGCNT                   = 00660078                  ERR$_UNIT                         = 00000008
DS$_ILLUNIT                     = 00660100                  FIND_END                            0000223D R     01
DS$_INSFMEM                     = 00660050                  HALT                              = 00000004  G
DS$_IPL2HI                      = 006600B8                  INITIALIZE                        = 00000001
DS$_IVADDR                      = 00660040                  INIT_FLAG                         = 00000004
DS$_IVVECT                      = 00660038                  INIT_SEQUENCE                       00001F59 R     01
DS$_KRNLSTK                     = 00660090                  INT_ERR                             00001F3D R     01
DS$_LOGIC                       = 00660070                  IPR                               = 00000006  G
DS$_MCHK                        = 00660088                  L_COUNT                             00000006 R     01
DS$_MMOFF                       = 00660058                  L_COUNT1                            0000000A R     01
DS$_NEEDUNIT                    = 006600F8                  L_COUNT3                            0000000E R     01
DS$_NODE                        = 00660128                  MANUAL                            = 00000001  G
DS$_NOPCS                       = 00660110                  MB_DIAGNOSE_CMD                     00000318 R     01
DS$_NORMAL                      = 00660001                  MB_END_PACKET_1                     00000660 R     01
DS$_NOSUPPORT                   = 006600B1                  MB_END_PACKET_10                    000006DE R     01
DS$_NOTDON                      = 00660030                  MB_END_PACKET_2                     0000066E R     01
DS$_NOTIMP                      = 006600B0                  MB_END_PACKET_3                     0000067C R     01
DS$_NULLSTR                     = 00660010                  MB_END_PACKET_4                     0000068A R     01
DS$_OVERFLOW                    = 00660008                  MB_END_PACKET_5                     00000698 R     01
DS$_POWER                       = 00660098                  MB_END_PACKET_6                     000006A6 R     01
DS$_PROGERR                     = 00660020                  MB_END_PACKET_7                     000006B4 R     01
DS$_SEVERE                      = 00660004                  MB_END_PACKET_8                     000006C2 R     01
DS$_TRANSL                      = 006600A0                  MB_END_PACKET_9                     000006D0 R     01
DS$_TRUNCATE                    = 00660028                  MB_INIT_CMD                         000002FC R     01
DS$_UNEXPINT                    = 006600D8                  MB_NOP_CMD                          0000030A R     01
DS$_VASFULL                     = 00660048                  MB_POS_1_CMD                        000005C6 R     01
DS$_WARNING                     = 00660000                  MB_READ_10_CMD                      0000044C R     01
DUMP_14                           00002227 R     01         MB_READ_11_CMD                      00000468 R     01
DUMP_PKT                          000021E7 R     01         MB_READ_12_CMD                      00000484 R     01
```

```
MB_READ_13_CMD                  000004A0 R      01        MREAD_SERVICE                   0000216C R      01
MB_READ_14_CMD                  000004BC R      01        MREAD_SERVICE2                  000021AC R      01
MB_READ_15_CMD                  000004D8 R      01        MRECEIVE_TU58                   000020C8 R      01
MB_READ_16_CMD                  000004F4 R      01        MSG_009                         00000002 R      03
MB_READ_17_CMD                  00000510 R      01        MSG_010                         00000002 R      06
MB_READ_18_CMD                  0000052C R      01        MSG_011                         00000002 R      07
MB_READ_19_CMD                  00000548 R      01        MSG_012                         00000002 R      08
MB_READ_1_CMD                   00000326 R      01        MSG_1                           00001AD6 R      01
MB_READ_20_CMD                  00000564 R      01        MSG_2                           00001AE6 R      01
MB_READ_21_CMD                  00000580 R      01        M_CLOCK_FLAG                  = 00000001
MB_READ_22_CMD                  0000059C R      01        M_DATA_FLAG                   = 00000004
MB_READ_23_CMD                  000005AA R      01        M_ENABLE                      = 00000040
MB_READ_24_CMD                  000005B8 R      01        M_END_FLAG                    = 00000008
MB_READ_2_CMD                   00000350 R      01        M_ERROR                       = 80000000
MB_READ_3_CMD                   0000036C R      01        M_INTERRUPT                   = 00000080
MB_READ_4_CMD                   00000388 R      01        M_RD_INT_FLAG                 = 00000004
MB_READ_5_CMD                   000003C0 R      01        M_RECEIVE_FLAG                = 00000002
MB_READ_6_CMD                   000003CE R      01        M_REPEAT_CHECK                = 00000004
MB_READ_7_CMD                   00000406 R      01        M_REPEAT_SEND                 = 00000002
MB_READ_8_CMD                   00000414 R      01        M_RUN                         = 00000001
MB_READ_9_CMD                   00000430 R      01        M_SDONE_FLAG                  = 00000002
MB_WRITE_10_CMD                 000003F8 R      01        M_SEND_FLAG                   = 00000010
MB_WRITE_11_CMD                 00000422 R      01        M_TRANSFER                    = 00000010
MB_WRITE_12_CMD                 0000043E R      01        NOT_USED                      = 00000000
MB_WRITE_13_CMD                 0000045A R      01        NO_INT_ERR                      00001F4B R      01
MB_WRITE_14_CMD                 00000476 R      01        NO_OPERATION                  = 00000000
MB_WRITE_15_CMD                 00000492 R      01        NO_READY                        00001BEB R      01
MB_WRITE_16_CMD                 000004AE R      01        PAR$M_ATDEF                   = 00000010
MB_WRITE_17_CMD                 000004CA R      01        PAR$M_ATHI                    = 00000008
MB_WRITE_18_CMD                 000004E6 R      01        PAR$M_ATLO                    = 00000004
MB_WRITE_19_CMD                 00000502 R      01        PAR$M_NODEF                   = 00000002
MB_WRITE_1_CMD                  00000334 R      01        PAR$V_ATDEF                   = 00000004
MB_WRITE_20_CMD                 0000051E R      01        PAR$V_ATHI                    = 00000003
MB_WRITE_21_CMD                 0000053A R      01        PAR$V_ATLO                    = 00000002
MB_WRITE_22_CMD                 00000556 R      01        PAR$V_NODEF                   = 00000001
MB_WRITE_23_CMD                 00000572 R      01        PAR$_BIN                      = 00000002
MB_WRITE_24_CMD                 0000058E R      01        PAR$_DEC                      = 0000000A
MB_WRITE_2_CMD                  00000342 R      01        PAR$_HEX                      = 00000010
MB_WRITE_3_CMD                  0000035E R      01        PAR$_NO                       = 00000000
MB_WRITE_4_CMD                  0000037A R      01        PAR$_OCT                      = 00000008
MB_WRITE_5_CMD                  00000396 R      01        PAR$_YES                      = 00000001
MB_WRITE_6_CMD                  000003A4 R      01        PCS                           = 00000008  G
MB_WRITE_7_CMD                  000003B2 R      01        POSITION                      = 00000005
MB_WRITE_8_CMD                  000003DC R      01        POWER_FAIL                    = 00000003  G
MB_WRITE_9_CMD                  000003EA R      01        PR$_ICCS                        ********   X    01
MCHK                          = 00000005  G               PR750$_CSRD                   = 0000001D
MCOMMAND_TBL_1                  000015B3 R      01        PR750$_CSRS                   = 0000001C
MCOMMAND_TBL_2                  0000161F R      01        PR750$_CSTD                   = 0000001F
MCOMMAND_TBL_3                  0000165B R      01        PR750$_CSTS                   = 0000001E
MCOMMAND_TBL_4                  0000176B R      01        PR750$_NICR                   = 00000019
MCOMMAND_TBL_5                  000017EB R      01        READ                          = 00000002
MINIT_SEQUENCE                  00001FCB R      01        READ_SERVICE                    00002164 R      01
MODIFIER_00                   = 00000000                  READ_SERVICE2                   000021A0 R      01
MODIFIER_01                   = 00000001                  RECEIVE_TU58                    00002083 R      01
MODIFIER_80                   = 00000080                  REC_ERR                         00001EB4 R      01
```

```
SCB$L_ACCESS                        00000020                  SYS$CANTIM                        00010268
SCB$L_ARITH                         00000034                  SYS$CLOSE                         000105B8
SCB$L_BREAK                         0000002C                  SYS$CLREF                         00010298
SCB$L_CHME                          00000044                  SYS$CONNECT                       000105C0
SCB$L_CHMK                          00000040                  SYS$DALLOC                        000102D8
SCB$L_CHMS                          00000048                  SYS$DASSGN                        000102E0
SCB$L_CHMU                          0000004C                  SYS$DISCONNECT                    000105D0
SCB$L_COMPAT                        00000030                  SYS$FAO                           00010350
SCB$L_CSRD                        = 000000F0                  SYS$FAOL                          00010358
SCB$L_CSTD                        = 000000F4                  SYS$GET                           00010580
SCB$L_KNLSTK                        00000008                  SYS$GETCHN                        000104C8
SCB$L_MACHCHK                       00000004                  SYS$GETTIM                        00010378
SCB$L_OPCCUS                        00000014                  SYS$LKWSET                        000103A0
SCB$L_OPCDEC                        00000010                  SYS$NUMTIM                        000103B8
SCB$L_POWER                         0000000C                  SYS$OPEN                          00010608
SCB$L_RADRMOD                       0000001C                  SYS$QIO                           000103C8
SCB$L_ROPRAND                       00000018                  SYS$QIOW                          00010200
SCB$L_RXDB                          000000F8                  SYS$READ                          00010590
SCB$L_SFTLVL1                       00000084                  SYS$READEF                        000103D0
SCB$L_SFTLVL10                      000000A8                  SYS$SETAST                        000103F8
SCB$L_SFTLVL11                      000000AC                  SYS$SETEF                         00010400
SCB$L_SFTLVL12                      000000B0                  SYS$SETIMR                        00010420
SCB$L_SFTLVL13                      000000B4                  SYS$SETPRI                        00010428
SCB$L_SFTLVL14                      000000B8                  SYS$SETPRT                        00010430
SCB$L_SFTLVL15                      000000BC                  SYS$SETRWM                        00010438
SCB$L_SFTLVL2                       00000088                  SYS$SETSWM                        00010448
SCB$L_SFTLVL3                       0000008C                  SYS$ULKPAG                        00010460
SCB$L_SFTLVL4                       00000090                  SYS$ULWSET                        00010468
SCB$L_SFTLVL5                       00000094                  SYS$UNWIND                        00010470
SCB$L_SFTLVL6                       00000098                  SYS$WAITFR                        00010478
SCB$L_SFTLVL7                       0000009C                  SYS$WFLAND                        00010488
SCB$L_SFTLVL8                       000000A0                  SYS$WFLOR                         00010490
SCB$L_SFTLVL9                       000000A4                  T10_S1                            00000131 RG     06
SCB$L_TBIT                          00000028                  T10_S10                           000014D8 RG     06
SCB$L_TIMER                         000000C0                  T10_S10_X                         00001854 R      06
SCB$L_TRANSL                        00000024                  T10_S11                           0000185F RG     06
SCB$L_TXDB                          000000FC                  T10_S11_X                         00001BB1 R      06
SCB$L_ZERO                          00000000                  T10_S12                           00001BBC RG     06
SEND_INT_ERR                        00001AC9 R      01        T10_S12_X                         00001DEB R      06
SEND_TU58                           0000203D R      01        T10_S13                           00001DF6 RG     06
SEP_FUNCTIONAL                    = 00000002                  T10_S13_X                         00001F86 R      06
SEP_REPAIR                        = 00000003                  T10_S14                           00001F91 RG     06
SEQUENCE_H                        = 00000000                  T10_S14_X                         00002105 R      06
SEQUENCE_L                        = 00000000                  T10_S1_X                          00000436 R      06
SIZ...                            = 00000001                  T10_S2                            00000441 RG     06
SUCCESS_CODE                      = 00000000                  T10_S2_X                          00000625 R      06
SUMMARY_STAT_H                    = 00000000                  T10_S3                            00000630 RG     06
SUMMARY_STAT_L                    = 00000000                  T10_S3_X                          00000786 R      06
SWITCH_00                         = 00000000                  T10_S4                            00000791 RG     06
SWITCH_08                         = 00000008                  T10_S4_X                          00000975 R      06
SYS$ALLOC                           00010238                  T10_S5                            00000980 RG     06
SYS$ASCTIM                          00010248                  T10_S5_X                          00000AD6 R      06
SYS$ASSIGN                          00010250                  T10_S6                            00000AE1 RG     06
SYS$BINTIM                          00010258                  T10_S6_X                          00000CBB R      06
SYS$CANCEL                          00010260                  T10_S7                            00000CC6 RG     06
```

```
T10_S7_X                        00000E0E R       06        TIMER_SERVICE                   00002150 R       01
T10_S8                          00000E19 RG      06        TU58                          = 00000002  G
T10_S8_X                        00001186 R       06        T_ACT_CONT                      00001C68 R       01
T10_S9                          00001191 RG      06        T_ACT_DATA                      00001C9B R       01
T10_S9_X                        000014CD R       06        T_ACT_END                       00001DB5 R       01
T11_S1                          00000135 RG      07        T_ACT_REC                       00001E18 R       01
T11_S1_X                        0000022D R       07        T_BAD_CONT                      00001BF8 R       01
T11_S2                          00000238 RG      07        T_BAD_DATA                      00001C37 R       01
T11_S2_X                        00000330 R       07        T_BAD_END                       00001C19 R       01
T11_S3                          0000033B RG      07        T_DUMP_14                       00001E42 R       01
T11_S3_X                        00000433 R       07        T_DUMP_HDR                      00001CBE R       01
T12_S1                          0000013D RG      08        T_DUMP_NOXOR                    00001CF2 R       01
T12_S10                         000014E4 RG      08        T_DUMP_XOR                      00001CDA R       01
T12_S10_X                       00001860 R       08        T_EMPTY_BUF                     00001D93 R       01
T12_S11                         0000186B RG      08        T_EXP_CONT                      00001C56 R       01
T12_S11_X                       00001BBD R       08        T_EXP_DATA                      00001C7A R       01
T12_S12                         00001BC8 RG      08        T_INIT1_ERR                     0000183B R       01
T12_S12_X                       00001DF7 R       08        T_INIT2_ERR                     00001865 R       01
T12_S13                         00001E02 RG      08        T_INT_ERR                       00001A2C R       01
T12_S13_X                       00001F8F R       08        T_IPL_ERR                       00001A04 R       01
T12_S14                         00001F9A RG      08        T_MSG_1                         00001AF6 R       01
T12_S14_X                       00002115 R       08        T_MSG_2                         00001B28 R       01
T12_S1_X                        00000442 R       08        T_NOP_ERR                       000018AD R       01
T12_S2                          0000044D RG      08        T_NO_INT_ERR                    00001A65 R       01
T12_S2_X                        00000631 R       08        T_NO_READY                      00001BB9 R       01
T12_S3                          0000063C RG      08        T_OVERWRITE                     00001B56 R       01
T12_S3_X                        00000792 R       08        T_RD_BLK_ERR                    000018EB R       01
T12_S4                          0000079D RG      08        T_RD_INT_ERR                    0000199F R       01
T12_S4_X                        00000981 R       08        T_RD_MOD_ERR                    0000197A R       01
T12_S5                          0000098C RG      08        T_RD_REC_ERR                    00001931 R       01
T12_S5_X                        00000AE2 R       08        T_REC_CONT                      00001D47 R       01
T12_S6                          00000AED RG      08        T_REC_DATA                      00001D68 R       01
T12_S6_X                        00000CC7 R       08        T_REC_END                       00001D2E R       01
T12_S7                          00000CD2 RG      08        T_REC_ERR                       00001D04 R       01
T12_S7_X                        00000E1A R       08        T_SEND_INT_ERR                  00001A89 R       01
T12_S8                          00000E25 RG      08        T_SLF_DIAG_ERR                  00001888 R       01
T12_S8_X                        00001192 R       08        T_UNREC_DATA                    00001DF4 R       01
T12_S9                          0000119D RG      08        T_WR_BLK_ERR                    000018C8 R       01
T12_S9_X                        000014D9 R       08        T_WR_INT_ERR                    000019D1 R       01
T9_S1                           00000129 RG      03        T_WR_MOD_ERR                    00001954 R       01
T9_S1_X                         00000221 R       03        T_WR_REC_ERR                    0000190D R       01
T9_S2                           0000022C RG      03        UNIT_0                        = 00000000
T9_S2_X                         00000324 R       03        V_CLOCK_FLAG                  = 00000000
T9_S3                           0000032F RG      03        V_DATA_FLAG                   = 00000002
T9_S3_X                         00000427 R       03        V_DONE                        = 00000007
TEMP5                           00000017 R       01        V_DONE_FLAG                   = 00000000
TEMP7                           0000001B R       01        V_END_FLAG                    = 00000003
TEST_009                        00000018 RG      03        V_RD_INT_FLAG                 = 00000002
TEST_009_X                      00000442 RG      03        V_READY                       = 00000007
TEST_010                        00000020 RG      06        V_RECEIVE_FLAG                = 00000001
TEST_010_X                      0000213A RG      06        V_REPEAT_CHECK                = 00000002
TEST_011                        00000024 RG      07        V_REPEAT_SEND                 = 00000001
TEST_011_X                      0000044E RG      07        V_RESPONSE_FLAG               = 00000004
TEST_012                        0000002C RG      08        V_RUN                         = 00000000
TEST_012_X                      0000214A RG      08        V_SDONE_FLAG                  = 00000001
```

```
V_SEND_FLAG                     = 00000004
V_TRANSFER                      = 00000004
WCS                             = 00000007  G
WRITE                           = 00000003
WRITE_SERVICE                     000021D4 R     01
XOFF_FLAG                       = 00000013
XON_FLAG                        = 00000011

                                                 +----------------+
                                                 ! Psect synopsis !
                                                 +----------------+

PSECT name                       Allocation          PSECT No.  Attributes
----------                       ----------          ---------  ----------
.  ABS  .                        00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
ECKAXG                           0000224D  ( 8781.)  01 (  1.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC PAGE
$ABS$                            00010610  (67088.)  02 (  2.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
TEST_009                         0000044A  ( 1098.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC PAGE
DISPATCH                         00000060  (   96.)  04 (  4.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD   NOWRT NOVEC LONG
ARGLIST                          00000198  (  408.)  05 (  5.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC LONG
TEST_010                         00002142  ( 8514.)  06 (  6.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC PAGE
TEST_011                         00000456  ( 1110.)  07 (  7.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC PAGE
TEST_012                         00002152  ( 8530.)  08 (  8.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC PAGE
$TSTCNT                          00000004  (    4.)  09 (  9.)  NOPIC   USR   OVR   REL   LCL NOSHR NOEXE   RD   NOWRT NOVEC LONG
```

```
                                             +------------------------+
                                             ! Symbol Cross Reference !
                                             +------------------------+

SYMBOL                         VALUE         DEFINITION      REFERENCES...
------                         -----         ----------      -------------
$$$                            =0000018C-R   9762   (94)     3775   (57)     3876   (58)     3978   (59)     4231   (61)
                                                             4439   (62)     4592   (63)     4728   (64)     4881   (65)
                                                             5014   (66)     5160   (67)     5350   (68)     5629   (69)
                                                             5896   (70)     6171   (71)     6396   (72)     6559   (73)
                                                             6674   (74)     6864   (77)     6965   (78)     7067   (79)
                                                             7320   (81)     7528   (82)     7681   (83)     7817   (84)
                                                             7970   (85)     8103   (86)     8249   (87)     8439   (88)
                                                             8718   (89)     8985   (90)     9260   (91)     9485   (92)
                                                             9648   (93)     9762   (94)
$$ARGS                         =0000000A     89     (2)      89     (2)
$$E                            =00000004     9762   (94)     3815   (57)     3917   (58)     4019   (59)     4358   (61)
                                                             4519   (62)     4648   (63)     4807   (64)     4935   (65)
                                                             5092   (66)     5213   (67)     5497   (68)     5771   (69)
                                                             6046   (70)     6315   (71)     6487   (72)     6616   (73)
                                                             6727   (74)     6904   (77)     7006   (78)     7108   (79)
                                                             7447   (81)     7608   (82)     7737   (83)     7896   (84)
                                                             8024   (85)     8181   (86)     8302   (87)     8586   (88)
                                                             8860   (89)     9135   (90)     9404   (91)     9576   (92)
                                                             9704   (93)     9816   (94)
$$M                            =00000004     9808   (94)     3668   (56)     3687   (56)     3695   (56)     3708   (56)
                                                             3786   (57)     3801   (57)     3810   (57)     3888   (58)
                                                             3903   (58)     3912   (58)     3990   (59)     4005   (59)
                                                             4014   (59)     4059   (60)     4078   (60)     4086   (60)
                                                             4099   (60)     4243   (61)     4254   (61)     4267   (61)
                                                             4288   (61)     4304   (61)     4314   (61)     4324   (61)
                                                             4341   (61)     4351   (61)     4451   (62)     4466   (62)
                                                             4476   (62)     4486   (62)     4504   (62)     4514   (62)
                                                             4604   (63)     4619   (63)     4630   (63)     4642   (63)
                                                             4740   (64)     4755   (64)     4765   (64)     4775   (64)
                                                             4792   (64)     4802   (64)     4893   (65)     4908   (65)
                                                             4919   (65)     4931   (65)     5027   (66)     5041   (66)
                                                             5051   (66)     5061   (66)     5077   (66)     5087   (66)
                                                             5172   (67)     5187   (67)     5197   (67)     5208   (67)
                                                             5366   (68)     5381   (68)     5391   (68)     5401   (68)
                                                             5419   (68)     5429   (68)     5450   (68)     5465   (68)
                                                             5476   (68)     5488   (68)     5645   (69)     5659   (69)
                                                             5669   (69)     5679   (69)     5696   (69)     5706   (69)
                                                             5726   (69)     5741   (69)     5751   (69)     5762   (69)
                                                             5912   (70)     5927   (70)     5937   (70)     5947   (70)
                                                             5969   (70)     5979   (70)     5994   (70)     6012   (70)
                                                             6023   (70)     6039   (70)     6187   (71)     6201   (71)
                                                             6211   (71)     6221   (71)     6241   (71)     6251   (71)
                                                             6265   (71)     6283   (71)     6293   (71)     6308   (71)
                                                             6410   (72)     6424   (72)     6434   (72)     6456   (72)
                                                             6472   (72)     6482   (72)     6574   (73)     6593   (73)
                                                             6601   (73)     6611   (73)     6685   (74)     6702   (74)
                                                             6719   (74)     6757   (76)     6776   (76)     6784   (76)
                                                             6797   (76)     6875   (77)     6890   (77)     6899   (77)
                                                             6977   (78)       6992   (78)       7001   (78)       7079   (79)
                                                             7094   (79)       7103   (79)       7148   (80)       7167   (80)
                                                             7175   (80)       7188   (80)       7332   (81)       7343   (81)
                                                             7356   (81)       7377   (81)       7393   (81)       7403   (81)
                                                             7413   (81)       7430   (81)       7440   (81)       7540   (82)
                                                             7555   (82)       7565   (82)       7575   (82)       7593   (82)
                                                             7603   (82)       7693   (83)       7708   (83)       7719   (83)
                                                             7731   (83)       7829   (84)       7844   (84)       7854   (84)
                                                             7864   (84)       7881   (84)       7891   (84)       7982   (85)
                                                             7997   (85)       8008   (85)       8020   (85)       8116   (86)
                                                             8130   (86)       8140   (86)       8150   (86)       8166   (86)
                                                             8176   (86)       8261   (87)       8276   (87)       8286   (87)
                                                             8297   (87)       8455   (88)       8470   (88)       8480   (88)
                                                             8490   (88)       8508   (88)       8518   (88)       8539   (88)
                                                             8554   (88)       8565   (88)       8577   (88)       8734   (89)
                                                             8748   (89)       8758   (89)       8768   (89)       8785   (89)
                                                             8795   (89)       8815   (89)       8830   (89)       8840   (89)
                                                             8851   (89)       9001   (90)       9016   (90)       9026   (90)
                                                             9036   (90)       9058   (90)       9068   (90)       9083   (90)
                                                             9101   (90)       9112   (90)       9128   (90)       9276   (91)
                                                             9290   (91)       9300   (91)       9310   (91)       9330   (91)
                                                             9340   (91)       9354   (91)       9372   (91)       9382   (91)
                                                             9397   (91)       9499   (92)       9513   (92)       9523   (92)
                                                             9545   (92)       9561   (92)       9571   (92)       9662   (93)
                                                             9681   (93)       9689   (93)       9699   (93)       9773   (94)
                                                             9790   (94)       9808   (94)
$$N                            =00000000     9808   (94)   #-3173   (53)     #-3178   (53)     #-3183   (53)     #-3200   (53)
                                                           #-3262   (53)       3263   (53)       3264   (53)     #-3269   (53)
                                                             3270   (53)       3271   (53)     #-3276   (53)     #-3290   (53)
                                                           #-3292   (53)       3294   (53)     #-3300   (53)     #-3306   (53)
                                                           #-3309   (53)       3310   (53)     #-3316   (53)     #-3321   (53)
                                                           #-3610   (54)       3614   (54)       3616   (54)       3626   (54)
                                                             3687   (56)       3695   (56)       3708   (56)       3786   (57)
                                                             3801   (57)       3810   (57)       3888   (58)       3903   (58)
                                                             3912   (58)       3990   (59)       4005   (59)       4014   (59)
                                                             4078   (60)       4086   (60)       4099   (60)       4243   (61)
                                                             4254   (61)       4267   (61)       4288   (61)       4304   (61)
                                                             4314   (61)       4324   (61)       4341   (61)       4351   (61)
                                                             4451   (62)       4466   (62)       4476   (62)       4486   (62)
                                                             4504   (62)       4514   (62)       4604   (63)       4619   (63)
                                                             4630   (63)       4642   (63)       4740   (64)       4755   (64)
                                                             4765   (64)       4775   (64)       4792   (64)       4802   (64)
                                                             4893   (65)       4908   (65)       4919   (65)       4931   (65)
                                                             5027   (66)       5041   (66)       5051   (66)       5061   (66)
                                                             5077   (66)       5087   (66)       5172   (67)       5187   (67)
                                                             5197   (67)       5208   (67)       5366   (68)       5381   (68)
                                                             5391   (68)       5401   (68)       5419   (68)       5429   (68)
                                                             5450   (68)       5465   (68)       5476   (68)       5488   (68)
                                                             5645   (69)       5659   (69)       5669   (69)       5679   (69)
                                                             5696   (69)       5706   (69)       5726   (69)       5741   (69)
                                                             5751   (69)       5762   (69)       5912   (70)       5927   (70)
                                                             5937   (70)       5947   (70)       5969   (70)       5979   (70)
                                                             5994   (70)       6012   (70)       6023   (70)       6039   (70)
                                                             6187   (71)       6201   (71)       6211   (71)       6221   (71)
                                                             6241   (71)       6251   (71)       6265   (71)       6283   (71)
```

```
                                                     6293    (71)    6308    (71)    6410    (72)    6424    (72)
                                                     6434    (72)    6456    (72)    6472    (72)    6482    (72)
                                                     6574    (73)    6593    (73)    6601    (73)    6611    (73)
                                                     6685    (74)    6702    (74)    6719    (74)    6776    (76)
                                                     6784    (76)    6797    (76)    6875    (77)    6890    (77)
                                                     6899    (77)    6977    (78)    6992    (78)    7001    (78)
                                                     7079    (79)    7094    (79)    7103    (79)    7167    (80)
                                                     7175    (80)    7188    (80)    7332    (81)    7343    (81)
                                                     7356    (81)    7377    (81)    7393    (81)    7403    (81)
                                                     7413    (81)    7430    (81)    7440    (81)    7540    (82)
                                                     7555    (82)    7565    (82)    7575    (82)    7593    (82)
                                                     7603    (82)    7693    (83)    7708    (83)    7719    (83)
                                                     7731    (83)    7829    (84)    7844    (84)    7854    (84)
                                                     7864    (84)    7881    (84)    7891    (84)    7982    (85)
                                                     7997    (85)    8008    (85)    8020    (85)    8116    (86)
                                                     8130    (86)    8140    (86)    8150    (86)    8166    (86)
                                                     8176    (86)    8261    (87)    8276    (87)    8286    (87)
                                                     8297    (87)    8455    (88)    8470    (88)    8480    (88)
                                                     8490    (88)    8508    (88)    8518    (88)    8539    (88)
                                                     8554    (88)    8565    (88)    8577    (88)    8734    (89)
                                                     8748    (89)    8758    (89)    8768    (89)    8785    (89)
                                                     8795    (89)    8815    (89)    8830    (89)    8840    (89)
                                                     8851    (89)    9001    (90)    9016    (90)    9026    (90)
                                                     9036    (90)    9058    (90)    9068    (90)    9083    (90)
                                                     9101    (90)    9112    (90)    9128    (90)    9276    (91)
                                                     9290    (91)    9300    (91)    9310    (91)    9330    (91)
                                                     9340    (91)    9354    (91)    9372    (91)    9382    (91)
                                                     9397    (91)    94      (2)     9499    (92)    9513    (92)
                                                     9523    (92)    9545    (92)    9561    (92)    9571    (92)
                                                     9662    (93)    9681    (93)    9689    (93)    9699    (93)
                                                     9773    (94)    9790    (94)    9808    (94)
$$S                             =FFFFFFFF     9822   (94)    3649    (56)    3668    (56)    3715    (57)    3775    (57)
                                                     3819    (58)    3876    (58)    3921    (59)    3978    (59)
                                                     4026    (60)    4059    (60)    4106    (61)    4231    (61)
                                                     4362    (62)    4439    (62)    4523    (63)    4592    (63)
                                                     4652    (64)    4728    (64)    4811    (65)    4881    (65)
                                                     4938    (66)    5014    (66)    5096    (67)    5160    (67)
                                                     5217    (68)    5350    (68)    5501    (69)    5629    (69)
                                                     5775    (70)    5896    (70)    6050    (71)    6171    (71)
                                                     6318    (72)    6396    (72)    6490    (73)    6559    (73)
                                                     6619    (74)    6674    (74)    6738    (76)    6757    (76)
                                                     6804    (77)    6864    (77)    6908    (78)    6965    (78)
                                                     7010    (79)    7067    (79)    7115    (80)    7148    (80)
                                                     7195    (81)    7320    (81)    7451    (82)    7528    (82)
                                                     7612    (83)    7681    (83)    7741    (84)    7817    (84)
                                                     7900    (85)    7970    (85)    8027    (86)    8103    (86)
                                                     8185    (87)    8249    (87)    8306    (88)    8439    (88)
                                                     8590    (89)    8718    (89)    8864    (90)    8985    (90)
                                                     9139    (91)    9260    (91)    9407    (92)    9485    (92)
                                                     9579    (93)    9648    (93)    9707    (94)    9762    (94)
$$T1                            =0000002C     89     (2)     89      (2)
$ENV                            =00000001     72     (2)
$ER                             =FFFFFFFF     9822   (94)   #-3687   (56)   #-3695   (56)   #-3708   (56)    3775    (57)
                                                    #-3786   (57)   #-3801   (57)   #-3810   (57)    3876    (58)
                                                    #-3888   (58)   #-3903   (58)   #-3912   (58)    3978    (59)
```

```
#-3990   (59)    #-4005   (59)    #-4014   (59)    #-4078   (60)
#-4086   (60)    #-4099   (60)      4231   (61)    #-4243   (61)
#-4254   (61)    #-4267   (61)    #-4288   (61)    #-4304   (61)
#-4314   (61)    #-4324   (61)    #-4341   (61)    #-4351   (61)
  4439   (62)    #-4451   (62)    #-4466   (62)    #-4476   (62)
#-4486   (62)    #-4504   (62)    #-4514   (62)      4592   (63)
#-4604   (63)    #-4619   (63)    #-4630   (63)    #-4642   (63)
  4728   (64)    #-4740   (64)    #-4755   (64)    #-4765   (64)
#-4775   (64)    #-4792   (64)    #-4802   (64)      4881   (65)
#-4893   (65)    #-4908   (65)    #-4919   (65)    #-4931   (65)
  5014   (66)    #-5027   (66)    #-5041   (66)    #-5051   (66)
#-5061   (66)    #-5077   (66)    #-5087   (66)      5160   (67)
#-5172   (67)    #-5187   (67)    #-5197   (67)    #-5208   (67)
  5350   (68)    #-5366   (68)    #-5381   (68)    #-5391   (68)
#-5401   (68)    #-5419   (68)    #-5429   (68)    #-5450   (68)
#-5465   (68)    #-5476   (68)    #-5488   (68)      5629   (69)
#-5645   (69)    #-5659   (69)    #-5669   (69)    #-5679   (69)
#-5696   (69)    #-5706   (69)    #-5726   (69)    #-5741   (69)
#-5751   (69)    #-5762   (69)      5896   (70)    #-5912   (70)
#-5927   (70)    #-5937   (70)    #-5947   (70)    #-5969   (70)
#-5979   (70)    #-5994   (70)    #-6012   (70)    #-6023   (70)
#-6039   (70)      6171   (71)    #-6187   (71)    #-6201   (71)
#-6211   (71)    #-6221   (71)    #-6241   (71)    #-6251   (71)
#-6265   (71)    #-6283   (71)    #-6293   (71)    #-6308   (71)
  6396   (72)    #-6410   (72)    #-6424   (72)    #-6434   (72)
#-6456   (72)    #-6472   (72)    #-6482   (72)      6559   (73)
#-6574   (73)    #-6593   (73)    #-6601   (73)    #-6611   (73)
  6674   (74)    #-6685   (74)    #-6702   (74)    #-6719   (74)
#-6776   (76)    #-6784   (76)    #-6797   (76)      6864   (77)
#-6875   (77)    #-6890   (77)    #-6899   (77)      6965   (78)
#-6977   (78)    #-6992   (78)    #-7001   (78)      7067   (79)
#-7079   (79)    #-7094   (79)    #-7103   (79)    #-7167   (80)
#-7175   (80)    #-7188   (80)      7320   (81)    #-7332   (81)
#-7343   (81)    #-7356   (81)    #-7377   (81)    #-7393   (81)
#-7403   (81)    #-7413   (81)    #-7430   (81)    #-7440   (81)
  7528   (82)    #-7540   (82)    #-7555   (82)    #-7565   (82)
#-7575   (82)    #-7593   (82)    #-7603   (82)      7681   (83)
#-7693   (83)    #-7708   (83)    #-7719   (83)    #-7731   (83)
  7817   (84)    #-7829   (84)    #-7844   (84)    #-7854   (84)
#-7864   (84)    #-7881   (84)    #-7891   (84)      7970   (85)
#-7982   (85)    #-7997   (85)    #-8008   (85)    #-8020   (85)
  8103   (86)    #-8116   (86)    #-8130   (86)    #-8140   (86)
#-8150   (86)    #-8166   (86)    #-8176   (86)      8249   (87)
#-8261   (87)    #-8276   (87)    #-8286   (87)    #-8297   (87)
  8439   (88)    #-8455   (88)    #-8470   (88)    #-8480   (88)
#-8490   (88)    #-8508   (88)    #-8518   (88)    #-8539   (88)
#-8554   (88)    #-8565   (88)    #-8577   (88)      8718   (89)
#-8734   (89)    #-8748   (89)    #-8758   (89)    #-8768   (89)
#-8785   (89)    #-8795   (89)    #-8815   (89)    #-8830   (89)
#-8840   (89)    #-8851   (89)      8985   (90)    #-9001   (90)
#-9016   (90)    #-9026   (90)    #-9036   (90)    #-9058   (90)
#-9068   (90)    #-9083   (90)    #-9101   (90)    #-9112   (90)
#-9128   (90)      9260   (91)    #-9276   (91)    #-9290   (91)
#-9300   (91)    #-9310   (91)    #-9330   (91)    #-9340   (91)
#-9354   (91)    #-9372   (91)    #-9382   (91)    #-9397   (91)
```

```
                                                                        9485   (92)   #-9499   (92)   #-9513   (92)   #-9523   (92)
                                                                      #-9545   (92)   #-9561   (92)   #-9571   (92)     9648   (93)
                                                                      #-9662   (93)   #-9681   (93)   #-9689   (93)   #-9699   (93)
                                                                        9762   (94)   #-9773   (94)   #-9790   (94)   #-9808   (94)
$MO                                      =00000001     9823   (94)      9823   (94)
$S$                                      =0000018C-R   9762   (94)      3775   (57)     3815   (57)     3876   (58)     3917   (58)
                                                                        3978   (59)     4019   (59)     4231   (61)     4358   (61)
                                                                        4439   (62)     4519   (62)     4592   (63)     4648   (63)
                                                                        4728   (64)     4807   (64)     4881   (65)     4935   (65)
                                                                        5014   (66)     5092   (66)     5160   (67)     5213   (67)
                                                                        5350   (68)     5497   (68)     5629   (69)     5771   (69)
                                                                        5896   (70)     6046   (70)     6171   (71)     6315   (71)
                                                                        6396   (72)     6487   (72)     6559   (73)     6616   (73)
                                                                        6674   (74)     6727   (74)     6864   (77)     6904   (77)
                                                                        6965   (78)     7006   (78)     7067   (79)     7108   (79)
                                                                        7320   (81)     7447   (81)     7528   (82)     7608   (82)
                                                                        7681   (83)     7737   (83)     7817   (84)     7896   (84)
                                                                        7970   (85)     8024   (85)     8103   (86)     8181   (86)
                                                                        8249   (87)     8302   (87)     8439   (88)     8586   (88)
                                                                        8718   (89)     8860   (89)     8985   (90)     9135   (90)
                                                                        9260   (91)     9404   (91)     9485   (92)     9576   (92)
                                                                        9648   (93)     9704   (93)     9762   (94)     9816   (94)
$ST                                      =00000000     9822   (94)      3668   (56)     3687   (56)     3695   (56)     3708   (56)
                                                                        3715   (57)     3775   (57)     3786   (57)     3801   (57)
                                                                        3810   (57)     3815   (57)     3819   (58)     3876   (58)
                                                                        3888   (58)     3903   (58)     3912   (58)     3917   (58)
                                                                        3921   (59)     3978   (59)     3990   (59)     4005   (59)
                                                                        4014   (59)     4019   (59)     4022   (59)     4059   (60)
                                                                        4078   (60)     4086   (60)     4099   (60)     4106   (61)
                                                                        4231   (61)     4243   (61)     4254   (61)     4267   (61)
                                                                        4279   (61)     4288   (61)     4304   (61)     4314   (61)
                                                                        4324   (61)     4341   (61)     4351   (61)     4358   (61)
                                                                        4362   (62)     4439   (62)     4451   (62)     4466   (62)
                                                                        4476   (62)     4486   (62)     4504   (62)     4514   (62)
                                                                        4519   (62)     4523   (63)     4592   (63)     4604   (63)
                                                                        4619   (63)     4630   (63)     4642   (63)     4648   (63)
                                                                        4652   (64)     4728   (64)     4740   (64)     4755   (64)
                                                                        4765   (64)     4775   (64)     4792   (64)     4802   (64)
                                                                        4807   (64)     4811   (65)     4881   (65)     4893   (65)
                                                                        4908   (65)     4919   (65)     4931   (65)     4935   (65)
                                                                        4938   (66)     5014   (66)     5027   (66)     5041   (66)
                                                                        5051   (66)     5061   (66)     5077   (66)     5087   (66)
                                                                        5092   (66)     5096   (67)     5160   (67)     5172   (67)
                                                                        5187   (67)     5197   (67)     5208   (67)     5213   (67)
                                                                        5217   (68)     5350   (68)     5366   (68)     5381   (68)
                                                                        5391   (68)     5401   (68)     5419   (68)     5429   (68)
                                                                        5450   (68)     5465   (68)     5476   (68)     5488   (68)
                                                                        5497   (68)     5501   (69)     5629   (69)     5645   (69)
                                                                        5659   (69)     5669   (69)     5679   (69)     5696   (69)
                                                                        5706   (69)     5726   (69)     5741   (69)     5751   (69)
                                                                        5762   (69)     5771   (69)     5775   (70)     5896   (70)
                                                                        5912   (70)     5927   (70)     5937   (70)     5947   (70)
                                                                        5969   (70)     5979   (70)     5994   (70)     6012   (70)
                                                                        6023   (70)     6039   (70)     6046   (70)     6050   (71)
                                                                        6171   (71)     6187   (71)     6201   (71)     6211   (71)
```

```
                                                                    6221    (71)      6241    (71)      6251    (71)      6265    (71)
                                                                    6283    (71)      6293    (71)      6308    (71)      6315    (71)
                                                                    6318    (72)      6396    (72)      6410    (72)      6424    (72)
                                                                    6434    (72)      6456    (72)      6472    (72)      6482    (72)
                                                                    6487    (72)      6490    (73)      6559    (73)      6574    (73)
                                                                    6593    (73)      6601    (73)      6611    (73)      6616    (73)
                                                                    6619    (74)      6674    (74)      6685    (74)      6702    (74)
                                                                    6719    (74)      6727    (74)      6733    (74)      6757    (76)
                                                                    6776    (76)      6784    (76)      6797    (76)      6804    (77)
                                                                    6864    (77)      6875    (77)      6890    (77)      6899    (77)
                                                                    6904    (77)      6908    (78)      6965    (78)      6977    (78)
                                                                    6992    (78)      7001    (78)      7006    (78)      7010    (79)
                                                                    7067    (79)      7079    (79)      7094    (79)      7103    (79)
                                                                    7108    (79)      7111    (79)      7148    (80)      7167    (80)
                                                                    7175    (80)      7188    (80)      7195    (81)      7320    (81)
                                                                    7332    (81)      7343    (81)      7356    (81)      7368    (81)
                                                                    7377    (81)      7393    (81)      7403    (81)      7413    (81)
                                                                    7430    (81)      7440    (81)      7447    (81)      7451    (82)
                                                                    7528    (82)      7540    (82)      7555    (82)      7565    (82)
                                                                    7575    (82)      7593    (82)      7603    (82)      7608    (82)
                                                                    7612    (83)      7681    (83)      7693    (83)      7708    (83)
                                                                    7719    (83)      7731    (83)      7737    (83)      7741    (84)
                                                                    7817    (84)      7829    (84)      7844    (84)      7854    (84)
                                                                    7864    (84)      7881    (84)      7891    (84)      7896    (84)
                                                                    7900    (85)      7970    (85)      7982    (85)      7997    (85)
                                                                    8008    (85)      8020    (85)      8024    (85)      8027    (86)
                                                                    8103    (86)      8116    (86)      8130    (86)      8140    (86)
                                                                    8150    (86)      8166    (86)      8176    (86)      8181    (86)
                                                                    8185    (87)      8249    (87)      8261    (87)      8276    (87)
                                                                    8286    (87)      8297    (87)      8302    (87)      8306    (88)
                                                                    8439    (88)      8455    (88)      8470    (88)      8480    (88)
                                                                    8490    (88)      8508    (88)      8518    (88)      8539    (88)
                                                                    8554    (88)      8565    (88)      8577    (88)      8586    (88)
                                                                    8590    (89)      8718    (89)      8734    (89)      8748    (89)
                                                                    8758    (89)      8768    (89)      8785    (89)      8795    (89)
                                                                    8815    (89)      8830    (89)      8840    (89)      8851    (89)
                                                                    8860    (89)      8864    (90)      8985    (90)      9001    (90)
                                                                    9016    (90)      9026    (90)      9036    (90)      9058    (90)
                                                                    9068    (90)      9083    (90)      9101    (90)      9112    (90)
                                                                    9128    (90)      9135    (90)      9139    (91)      9260    (91)
                                                                    9276    (91)      9290    (91)      9300    (91)      9310    (91)
                                                                    9330    (91)      9340    (91)      9354    (91)      9372    (91)
                                                                    9382    (91)      9397    (91)      9404    (91)      9407    (92)
                                                                    9485    (92)      9499    (92)      9513    (92)      9523    (92)
                                                                    9545    (92)      9561    (92)      9571    (92)      9576    (92)
                                                                    9579    (93)      9648    (93)      9662    (93)      9681    (93)
                                                                    9689    (93)      9699    (93)      9704    (93)      9707    (94)
                                                                    9762    (94)      9773    (94)      9790    (94)      9808    (94)
                                                                    9816    (94)      9822    (94)
$TN                                   =0000000C     9823    (94)    3649    (56)      3668    (56)      3687    (56)      3695    (56)
                                                                    3708    (56)      3715    (57)      3786    (57)      3801    (57)
                                                                    3810    (57)      3819    (58)      3888    (58)      3903    (58)
                                                                    3912    (58)      3921    (59)      3990    (59)      4005    (59)
                                                                    4014    (59)      4022    (59)      4026    (60)      4059    (60)
                                                                    4078    (60)      4086    (60)      4099    (60)      4106    (61)
```

```
                                                     4243     (61)      4254     (61)      4267     (61)      4281     (61)
                                                     4288     (61)      4304     (61)      4314     (61)      4324     (61)
                                                     4341     (61)      4351     (61)      4362     (62)      4451     (62)
                                                     4466     (62)      4476     (62)      4486     (62)      4504     (62)
                                                     4514     (62)      4523     (63)      4604     (63)      4619     (63)
                                                     4630     (63)      4642     (63)      4652     (64)      4740     (64)
                                                     4755     (64)      4765     (64)      4775     (64)      4792     (64)
                                                     4802     (64)      4811     (65)      4893     (65)      4908     (65)
                                                     4919     (65)      4931     (65)      4938     (66)      5027     (66)
                                                     5041     (66)      5051     (66)      5061     (66)      5077     (66)
                                                     5087     (66)      5096     (67)      5172     (67)      5187     (67)
                                                     5197     (67)      5208     (67)      5217     (68)      5366     (68)
                                                     5381     (68)      5391     (68)      5401     (68)      5419     (68)
                                                     5429     (68)      5450     (68)      5465     (68)      5476     (68)
                                                     5488     (68)      5501     (69)      5645     (69)      5659     (69)
                                                     5669     (69)      5679     (69)      5696     (69)      5706     (69)
                                                     5726     (69)      5741     (69)      5751     (69)      5762     (69)
                                                     5775     (70)      5912     (70)      5927     (70)      5937     (70)
                                                     5947     (70)      5969     (70)      5979     (70)      5994     (70)
                                                     6012     (70)      6023     (70)      6039     (70)      6050     (71)
                                                     6187     (71)      6201     (71)      6211     (71)      6221     (71)
                                                     6241     (71)      6251     (71)      6265     (71)      6283     (71)
                                                     6293     (71)      6308     (71)      6318     (72)      6410     (72)
                                                     6424     (72)      6434     (72)      6456     (72)      6472     (72)
                                                     6482     (72)      6490     (73)      6574     (73)      6593     (73)
                                                     6601     (73)      6611     (73)      6619     (74)      6685     (74)
                                                     6702     (74)      6719     (74)      6733     (74)      6738     (76)
                                                     6757     (76)      6776     (76)      6784     (76)      6797     (76)
                                                     6804     (77)      6875     (77)      6890     (77)      6899     (77)
                                                     6908     (78)      6977     (78)      6992     (78)      7001     (78)
                                                     7010     (79)      7079     (79)      7094     (79)      7103     (79)
                                                     7111     (79)      7115     (80)      7148     (80)      7167     (80)
                                                     7175     (80)      7188     (80)      7195     (81)      7332     (81)
                                                     7343     (81)      7356     (81)      7370     (81)      7377     (81)
                                                     7393     (81)      7403     (81)      7413     (81)      7430     (81`
                                                     7440     (81)      7451     (82)      7540     (82)      7555     (82)
                                                     7565     (82)      7575     (82)      7593     (82)      7603     (82)
                                                     7612     (83)      7693     (83)      7708     (83)      7719     (83)
                                                     7731     (83)      7741     (84)      7829     (84)      7844     (84)
                                                     7854     (84)      7864     (84)      7881     (84)      7891     (84)
                                                     7900     (85)      7982     (85)      7997     (85)      8008     (85)
                                                     8020     (85)      8027     (86)      8116     (86)      8130     (86)
                                                     8140     (86)      8150     (86)      8166     (86)      8176     (86)
                                                     8185     (87)      8261     (87)      8276     (87)      8286     (87)
                                                     8297     (87)      8306     (88)      8455     (88)      8470     (88)
                                                     8480     (88)      8490     (88)      8508     (88)      8518     (88)
                                                     8539     (88)      8554     (88)      8565     (88)      8577     (88)
                                                     8590     (89)      8734     (89)      8748     (89)      8758     (89)
                                                     8768     (89)      8785     (89)      8795     (89)      8815     (89)
                                                     8830     (89)      8840     (89)      8851     (89)      8864     (90)
                                                     9001     (90)      9016     (90)      9026     (90)      9036     (90)
                                                     9058     (90)      9068     (90)      9083     (90)      9101     (90)
                                                     9112     (90)      9128     (90)      9139     (91)      9276     (91)
                                                     9290     (91)      9300     (91)      9310     (91)      9330     (91)
                                                     9340     (91)      9354     (91)      9372     (91)      9382     (91)
                                                     9397   (91)      9407   (92)      9499   (92)      9513   (92)
                                                     9523   (92)      9545   (92)      9561   (92)      9571   (92)
                                                     9579   (93)      9662   (93)      9681   (93)      9689   (93)
                                                     9699   (93)      9707   (94)      9773   (94)      9790   (94)
                                                     9808   (94)      9822   (94)      9823   (94)
ALL                            =00000009  94     (2)     3668   (56)      4059   (60)      6757   (76)      7148   (80)
A_BUFFER_BEGIN                  00000000-XR              3358   (54)      3392   (54)      3675   (56)      3684   (56)
                                                     3778   (57)      3792   (57)      3796   (57)      3880   (58)
                                                     3894   (58)      3898   (58)      3982   (59)      3996   (59)
                                                     4000   (59)      4066   (60)      4075   (60)      4234   (61)
                                                     4249   (61)      4261   (61)      4273   (61)      4294   (61)
                                                     4299   (61)      4330   (61)      4335   (61)      4442   (62)
                                                     4458   (62)      4462   (62)    #-4491   (62)      4495   (62)
                                                     4499   (62)      4595   (63)      4610   (63)      4613   (63)
                                                     4731   (64)      4747   (64)      4751   (64)    #-4779   (64)
                                                     4783   (64)      4787   (64)      4884   (65)      4899   (65)
                                                     4902   (65)      5018   (66)      5033   (66)      5037   (66)
                                                     5068   (66)      5072   (66)      5163   (67)      5178   (67)
                                                     5181   (67)      5357   (68)      5373   (68)      5377   (68)
                                                   #-5406   (68)      5410   (68)      5414   (68)      5441   (68)
                                                     5456   (68)      5459   (68)      5636   (69)      5651   (69)
                                                     5655   (69)    #-5684   (69)      5687   (69)      5691   (69)
                                                     5717   (69)      5732   (69)      5735   (69)      5903   (70)
                                                     5919   (70)      5923   (70)    #-5952   (70)      5960   (70)
                                                     5964   (70)      5985   (70)      6000   (70)      6006   (70)
                                                     6178   (71)      6193   (71)      6197   (71)    #-6226   (71)
                                                     6232   (71)      6236   (71)      6256   (71)      6271   (71)
                                                     6277   (71)      6401   (72)      6416   (72)      6420   (72)
                                                     6463   (72)      6467   (72)      6563   (73)      6568   (73)
                                                     6586   (73)      6764   (76)      6773   (76)      6867   (77)
                                                     6881   (77)      6885   (77)      6969   (78)      6983   (78)
                                                     6987   (78)      7071   (79)      7085   (79)      7089   (79)
                                                     7155   (80)      7164   (80)      7323   (81)      7338   (81)
                                                     7350   (81)      7362   (81)      7383   (81)      7388   (81)
                                                     7419   (81)      7424   (81)      7531   (82)      7547   (82)
                                                     7551   (82)    #-7580   (82)      7584   (82)      7588   (82)
                                                     7684   (83)      7699   (83)      7702   (83)      7820   (84)
                                                     7836   (84)      7840   (84)    #-7868   (84)      7872   (84)
                                                     7876   (84)      7973   (85)      7988   (85)      7991   (85)
                                                     8107   (86)      8122   (86)      8126   (86)      8157   (86)
                                                     8161   (86)      8252   (87)      8267   (87)      8270   (87)
                                                     8446   (88)      8462   (88)      8466   (88)    #-8495   (88)
                                                     8499   (88)      8503   (88)      8530   (88)      8545   (88)
                                                     8548   (88)      8725   (89)      8740   (89)      8744   (89)
                                                   #-8773   (89)      8776   (89)      8780   (89)      8806   (89)
                                                     8821   (89)      8824   (89)      8992   (90)      9008   (90)
                                                     9012   (90)    #-9041   (90)      9049   (90)      9053   (90)
                                                     9074   (90)      9089   (90)      9095   (90)      9267   (91)
                                                     9282   (91)      9286   (91)    #-9315   (91)      9321   (91)
                                                     9325   (91)      9345   (91)      9360   (91)      9366   (91)
                                                     9490   (92)      9505   (92)      9509   (92)      9552   (92)
                                                     9556   (92)      9652   (93)      9656   (93)      9674   (93)
                                                     9795   (94)
BASE_009                        00000000-R  3649   (56)      3668   (56)
BASE_010                        00000000-R  4026   (60)      4059   (60)
```

```
BASE_011                        00000000-R    6738     (76)       6757     (76)
BASE_012                        00000000-R    7115     (80)       7148     (80)
BIT...                         =00000005      92       (2)        72       (2)        90       (2)        92       (2)
BLOCK_00                       =00000000      287      (4)        1206     (21)       1207     (21)       1226     (21)       1243     (22)
                                                                  1260     (22)       1261     (22)       1277     (22)       1278     (22)
                                                                  1294     (22)       1295     (22)       1312     (23)       1329     (23)
                                                                  1346     (23)       1362     (24)       1379     (24)       1396     (24)
                                                                  1412     (25)       1429     (25)       1446     (25)       1463     (26)
                                                                  1480     (26)       1497     (26)       1514     (27)       1531     (27)
                                                                  1565     (28)       1582     (28)       1615     (29)       1616     (29)
                                                                  1632     (29)       1633     (29)       1683     (30)       1700     (30)
                                                                  1752     (31)       1769     (32)       1819     (33)       1836     (33)
                                                                  2086     (38)       2087     (38)       2106     (38)       2123     (39)
                                                                  380      (5)        381      (5)        397      (5)        398      (5)
                                                                  414      (5)        415      (5)        432      (6)        449      (6)
                                                                  466      (6)        482      (7)        499      (7)        516      (7)
                                                                  532      (8)        549      (8)        566      (8)        583      (9)
                                                                  600      (9)        617      (9)        634      (10)       651      (10)
                                                                  685      (11)       702      (11)       735      (12)       736      (12)
                                                                  752      (12)       753      (12)       803      (13)       820      (13)
                                                                  872      (14)       889      (15)       939      (16)       956      (16)
BLOCK 01                       =00000001      288      (4)        1650     (29)       1667     (30)       1684     (30)       1701     (30)
                                                                  1717     (31)       1718     (31)       1734     (31)       1735     (31)
                                                                  1786     (32)       1803     (32)       770      (12)       787      (13)
                                                                  804      (13)       821      (13)       837      (14)       838      (14)
                                                                  854      (14)       855      (14)       906      (15)       923      (15)
BLOCK_02                       =00000002      289      (4)        1107     (19)       1124     (19)       1174     (20)       1190     (21)
                                                                  1548     (27)       1599     (28)       1853     (33)       1869     (34)
                                                                  1987     (36)       2004     (36)       2054     (37)       2070     (38)
                                                                  668      (10)       719      (11)       973      (16)       989      (17)
BLOCK_04                       =00000004      290      (4)        1005     (17)       1022     (17)       1852     (33)       1868     (34)
                                                                  1885     (34)       1902     (34)       972      (16)       988      (17)
BLOCK_05                       =00000005      291      (4)        1072     (18)       1089     (19)       1952     (35)       1969     (36)
BLOCK_06                       =00000006      292      (4)        1006     (17)       1023     (17)       1038     (18)       1039     (18)
                                                                  1055     (18)       1056     (18)       1073     (18)       1090     (19)
                                                                  1140     (20)       1157     (20)       1173     (20)       1189     (21)
                                                                  1311     (23)       1328     (23)       1886     (34)       1903     (34)
                                                                  1918     (35)       1919     (35)       1935     (35)       1936     (35)
                                                                  1953     (35)       1970     (36)       2020     (37)       2037     (37)
                                                                  2053     (37)       2069     (38)       431      (6)        448      (6)
BLOCK_0A                        0000000A      293      (4)        1345     (23)       1361     (24)       465      (6)        481      (7)
BLOCK_0C                       =0000000C      294      (4)        1411     (25)       1428     (25)       531      (8)        548      (8)
BLOCK_0D                       =0000000D      295      (4)        1530     (27)       1547     (27)       1564     (28)       1581     (28)
                                                                  1598     (28)       650      (10)       667      (10)       684      (11)
                                                                  701      (11)       718      (11)
BLOCK_0E                       =0000000E      296      (4)        1378     (24)       1395     (24)       498      (7)        515      (7)
BLOCK_0F                       =0000000F      297      (4)        1445     (25)       1479     (26)       1496     (26)       565      (8)
                                                                  599      (9)        616      (9)
BLOCK 15                       -00000015      298      (4)        1225     (21)       1242     (22)       1751     (31)       1768     (32)
                                                                  1785     (32)       1802     (32)       2105     (38)       2122     (39)
                                                                  871      (14)       888      (15)       905      (15)       922      (15)
BLOCK_5A                        0000005A      299      (4)        1106     (19)       1123     (19)       1139     (20)       1156     (20)
                                                                  1986     (36)       2003     (36)       2019     (37)       2036     (37)
BLOCK_80                       -00000080      300      (4)        1649     (29)       1666     (30)       1818     (33)       1835     (33)
                                                                  769      (12)       786      (13)       938      (16)       955      (16)
```

```
BLOCK_8F                        =0000008F     301     (4)      1462    (26)      1513    (27)      582     (9)       633     (10)
BUFFER_SIZE                     =00000600     192     (2)     #-5358   (68)     #-5442   (68)     #-5637   (69)     #-5718   (69)
                                                              #-5904   (70)     #-5986   (70)     #-6179   (71)     #-6257   (71)
                                                              #-8447   (88)     #-8531   (88)     #-8726   (89)     #-8807   (89)
                                                              #-8993   (90)     #-9075   (90)     #-9268   (91)     #-9346   (91)
BUF_LIM                          00000013-R   165     (2)     #-3450   (54)     #-3477   (54)     #-3676   (56)     #-4067   (60)
                                                              #-6765   (76)     #-7156   (80)
BYTE_COUNT_00                   =00000000     305     (4)      1004    (17)      1021    (17)      1037    (18)      1054    (18)
                                                               1071    (18)      1088    (19)      1105    (19)      1122    (19)
                                                               1138    (20)      1155    (20)      1172    (20)      1188    (21)
                                                               1204    (21)      1223    (21)      1240    (22)      1241    (22)
                                                               1258    (22)      1259    (22)      1275    (22)      1276    (22)
                                                               1292    (22)      1293    (22)      1310    (23)      1327    (23)
                                                               1343    (23)      1359    (24)      1377    (24)      1394    (24)
                                                               1409    (25)      1426    (25)      1443    (25)      1460    (26)
                                                               1478    (26)      1494    (26)      1511    (27)      1529    (27)
                                                               1546    (27)      1563    (28)      1580    (28)      1597    (28)
                                                               1613    (29)      1630    (29)      1647    (29)      1664    (30)
                                                               1681    (30)      1698    (30)      1732    (31)      1749    (31)
                                                               1766    (32)      1783    (32)      1800    (32)      1833    (33)
                                                               1850    (33)      1866    (34)      1884    (34)      1901    (34)
                                                               1917    (35)      1934    (35)      1951    (35)      1968    (36)
                                                               1985    (36)      2002    (36)      2018    (37)      2035    (37)
                                                               2052    (37)      2068    (38)      2084    (38)      2103    (38)
                                                               2120    (39)      2121    (39)      2144    (40)      2145    (40)
                                                               2162    (40)      2179    (40)      2195    (41)      2213    (41)
                                                               2230    (41)      2246    (42)      2279    (42)      2297    (43)
                                                               2314    (43)      2315    (43)      2332    (43)      2349    (43)
                                                               2365    (44)      2383    (44)      2400    (44)      2416    (45)
                                                               2449    (45)      2467    (46)      378     (5)       379     (5)
                                                               395     (5)       396     (5)       412     (5)       413     (5)
                                                               430     (6)       447     (6)       463     (6)       479     (7)
                                                               497     (7)       514     (7)       529     (8)       546     (8)
                                                               563     (8)       580     (9)       598     (9)       614     (9)
                                                               631     (10)      649     (10)      666     (10)      683     (11)
                                                               700     (11)      717     (11)      733     (12)      750     (12)
                                                               767     (12)      784     (13)      801     (13)      818     (13)
                                                               852     (14)      869     (14)      886     (15)      903     (15)
                                                               920     (15)      953     (16)      970     (16)      986     (17)
BYTE_COUNT_01                   =00000001     306     (4)      1477    (26)      1562    (28)      1716    (31)      1817    (33)
                                                               1851    (33)      1867    (34)      2229    (41)      2264    (42)
                                                               2280    (42)      2399    (44)      2434    (45)      2450    (45)
                                                               2514    (47)      597     (9)       682     (11)      836     (14)
                                                               937     (16)      971     (16)      987     (17)
BYTE_COUNT_02                   =00000002     307     (4)      1205    (21)      1224    (21)      1344    (23)      1360    (24)
                                                               1410    (25)      1427    (25)      1444    (25)      1461    (26)
                                                               1495    (26)      1512    (27)      1648    (29)      1665    (30)
                                                               1682    (30)      1699    (30)      1733    (31)      1750    (31)
                                                               1767    (32)      1784    (32)      1801    (32)      1834    (33)
                                                               2085    (38)      2104    (38)      2196    (41)      2366    (44)
                                                               464     (6)       480     (7)       530     (8)       547     (8)
                                                               564     (8)       581     (9)       615     (9)       632     (10)
                                                               768     (12)      785     (13)      802     (13)      819     (13)
                                                               853     (14)      870     (14)      887     (15)      904     (15)
                                                               921     (15)      954     (16)
```

```
BYTE_COUNT_04                  =00000004     308     (4)      1614    (29)      1631    (29)      2247    (42)      2417    (45)
                                                              734     (12)      751     (12)
BYTE_COUNT_0A                  =0000000A     309     (4)      1013    (17)      1029    (18)      1046    (18)      1063    (18)
                                                              1080    (19)      1097    (19)      1114    (19)      1130    (20)
                                                              1147    (20)      1164    (20)      1180    (21)      1197    (21)
                                                              1216    (21)      1233    (22)      1251    (22)      1268    (22)
                                                              1285    (22)      1302    (23)      1319    (23)      1336    (23)
                                                              1352    (24)      1369    (24)      1386    (24)      1402    (25)
                                                              1419    (25)      1436    (25)      1453    (26)      1470    (26)
                                                              1487    (26)      1504    (27)      1521    (27)      1538    (27)
                                                              1555    (28)      1572    (28)      1589    (28)      1606    (29)
                                                              1623    (29)      1640    (29)      1657    (30)      1674    (30)
                                                              1691    (30)      1708    (31)      1725    (31)      1742    (31)
                                                              1759    (32)      1776    (32)      1793    (32)      1809    (33)
                                                              1826    (33)      1843    (33)      1859    (34)      1876    (34)
                                                              1893    (34)      1909    (35)      1926    (35)      1943    (35)
                                                              1960    (36)      1977    (36)      1994    (36)      2010    (37)
                                                              2027    (37)      2044    (37)      2060    (38)      2077    (38)
                                                              2096    (38)      2113    (39)      2137    (40)      2154    (40)
                                                              2171    (40)      2188    (41)      2205    (41)      2222    (41)
                                                              2239    (42)      2256    (42)      2272    (42)      2289    (43)
                                                              2307    (43)      2324    (43)      2341    (43)      2358    (44)
                                                              2375    (44)      2392    (44)      2409    (45)      2426    (45)
                                                              2442    (45)      2459    (46)      371     (5)       388     (5)
                                                              405     (5)       422     (6)       439     (6)       456     (6)
                                                              472     (7)       489     (7)       506     (7)       522     (8)
                                                              539     (8)       556     (8)       573     (9)       590     (9)
                                                              607     (9)       624     (10)      641     (10)      658     (10)
                                                              675     (11)      692     (11)      709     (11)      726     (12)
                                                              743     (12)      760     (12)      777     (13)      794     (13)
                                                              811     (13)      828     (14)      845     (14)      862     (14)
                                                              879     (15)      896     (15)      913     (15)      929     (16)
                                                              946     (16)      963     (16)      979     (17)      996     (17)
BYTE_COUNT 12                  =00000012     311     (4)      1326    (23)      446     (6)
BYTE_COUNT 1A                  =0000001A     310     (4)      2178    (40)      2348    (43)      2478    (47)
BYTE_COUNT_64                  =00000064     312     (4)      1070    (18)      1171    (20)      1950    (35)      2051    (37)
                                                              2296    (43)      2466    (46)      2653    (50)      2685    (51)
BYTE_COUNT_74                  =00000074     313     (4)      2570    (48)      2611    (49)
BYTE_COUNT_80                  =00000080     314     (4)      1003    (17)      1020    (17)      1036    (18)      1053    (18)
                                                              1087    (19)      1104    (19)      1121    (19)      1137    (20)
                                                              1154    (20)      1187    (21)      1309    (23)      1376    (24)
                                                              1393    (24)      1528    (27)      1545    (27)      1579    (28)
                                                              1596    (28)      1883    (34)      1900    (34)      1916    (35)
                                                              1933    (35)      1967    (36)      1984    (36)      2001    (36)
                                                              2017    (37)      2034    (37)      2067    (38)      2161    (40)
                                                              2212    (41)      2331    (43)      2382    (44)      2498    (47)
                                                              2506    (47)      2522    (47)      2530    (48)      2538    (48)
                                                              2546    (48)      2554    (48)      2562    (48)      2578    (49)
                                                              2587    (49)      2595    (49)      2603    (49)      2620    (49)
                                                              2629    (50)      2637    (50)      2645    (50)      2661    (50)
                                                              2669    (50)      2677    (51)      2693    (51)      429     (6)
                                                              496     (7)       513     (7)       648     (10)      665     (10)
                                                              699     (11)      716     (11)
BYTE COUNT F4                  =000000F4     315     (4)      1715    (31)      1816    (33)      2263    (42)      2433    (45)
                                                              835     (14)      936     (16)
```

```
B_CONTINUE                       00000023-R   361      (5)        3472     (54)      3547     (54)      3577     (54)      3683     (56)
                                                                   4074     (60)      4298     (61)      4461     (62)      4750     (64)
                                                                   5036     (66)      5376     (68)      5654     (69)      5922     (70)
                                                                   6196     (71)      6419     (72)      6772     (76)      7163     (80)
                                                                   7387     (81)      7550     (82)      7839     (84)      8125     (86)
                                                                   8465     (88)      8743     (89)      9011     (90)      9285     (91)
                                                                   9508     (92)
B_COUNT                          00000003-R   158      (2)      #-4456     (62)    #-4492     (62)    #-4745     (64)    #-4780     (64)
                                                                 #-5371     (68)    #-5407     (68)    #-5471     (68)    #-5482     (68)
                                                                 #-5492     (68)    #-5917     (70)    #-5953     (70)    #-5955     (70)
                                                                 #-7545     (82)    #-7581     (82)    #-7834     (84)    #-7869     (84)
                                                                 #-8460     (88)    #-8496     (88)    #-8560     (88)    #-8571     (88)
                                                                 #-8581     (88)    #-9006     (90)    #-9042     (90)    #-9044     (90)
B_COUNT1                         00000004-R   159      (2)      #-5351     (68)    #-5434     (68)    #-5438     (68)    #-5631     (69)
                                                                 #-5711     (69)    #-5714     (69)    #-5766     (69)    #-5899     (70)
                                                                 #-6043     (70)    #-6174     (71)    #-6312     (71)    #-8440     (88)
                                                                 #-8523     (88)    #-8527     (88)    #-8720     (89)    #-8800     (89)
                                                                 #-8803     (89)    #-8855     (89)    #-8988     (90)    #-9132     (90)
                                                                 #-9263     (91)    #-9401     (91)
B_COUNT2                         00000005-R   160      (2)      #-5352     (68)    #-5407     (68)    #-5433     (68)    #-5435     (68)
                                                                 #-8441     (88)    #-8496     (88)    #-8522     (88)    #-8524     (88)
B_DATA_PKT_1                     000006EC-R   2476     (47)       4271     (61)      4320     (61)      7360     (81)      7409     (81)
B_DATA_PKT_10                    00000AAB-R   2560     (48)
B_DATA_PKT_11                    00000B2F-R   2568     (48)
B_DATA_PKT_12                    00000BA7-R   2576     (49)
B_DATA_PKT_13                    00000C2B-R   2585     (49)       2814     (52)      2815     (52)      2816     (52)      2817     (52)
                                                                   2821     (52)      2822     (52)      2823     (52)      2824     (52)
                                                                   2894     (52)      2895     (52)      2896     (52)      2897     (52)
                                                                   3025     (52)      3026     (52)      3027     (52)      3028     (52)
                                                                   3032     (52)      3033     (52)      3034     (52)      3035     (52)
                                                                   3105     (52)      3106     (52)      3107     (52)      3108     (52)
B_DATA_PKT_14                    00000CAF-R   2593     (49)       2828     (52)      2829     (52)      2830     (52)      2831     (52)
                                                                   2835     (52)      2836     (52)      2837     (52)      2838     (52)
                                                                   3039     (52)      3040     (52)      3041     (52)      3042     (52)
                                                                   3046     (52)      3047     (52)      3048     (52)      3049     (52)
B_DATA_PKT_15                    00000D33-R   2601     (49)
B_DATA_PKT_16                    00000DB7-R   2609     (49)
B_DATA_PKT_17                    00000E3B-R   2618     (49)
B_DATA_PKT_18                    00000EBF-R   2627     (50)       2844     (52)      2845     (52)      2849     (52)      2850     (52)
                                                                   3055     (52)      3056     (52)      3060     (52)      3061     (52)
B_DATA_PKT_19                    00000F43-R   2635     (50)       2854     (52)      2858     (52)      3065     (52)      3069     (52)
B_DATA_PKT_2                     0000070A-R   2496     (47)       4482     (62)      4614     (63)      4771     (64)      4903     (65)
                                                                   5057     (66)      5182     (67)      6442     (72)      6587     (73)
                                                                   7571     (82)      7703     (83)      7860     (84)      7992     (85)
                                                                   8146     (86)      8271     (87)      9531     (92)      9675     (93)
B_DATA_PKT_20                    00000FC7-R   2643     (50)       2862     (52)      2866     (52)      3073     (52)      3077     (52)
B_DATA_PKT_21                    0000104B-R   2651     (50)
B_DATA_PKT_22                    000010B3-R   2659     (50)
B_DATA_PKT_23                    00001137-R   2667     (50)       2870     (52)      2874     (52)      3081     (52)      3085     (52)
B_DATA_PKT_24                    000011BB-R   2675     (51)       2878     (52)      2882     (52)      3089     (52)      3093     (52)
B_DATA_PKT_25                    0000123F-R   2683     (51)
B_DATA_PKT_26                    000012A7-R   2691     (51)
B_DATA_PKT_3                     0000078E-R   2504     (47)       2708     (52)      2709     (52)      2710     (52)      2711     (52)
                                                                   2715     (52)      2716     (52)      2717     (52)      2718     (52)
                                                                   2733     (52)      2734     (52)      2735     (52)      2736     (52)
```

```
                                                            2742    (52)     2746    (52)     2758    (52)     2919    (52)
                                                            2920    (52)     2921    (52)     2922    (52)     2926    (52)
                                                            2927    (52)     2928    (52)     2929    (52)     2944    (52)
                                                            2945    (52)     2946    (52)     2947    (52)     2953    (52)
                                                            2957    (52)     2969    (52)
B_DATA_PKT_4                   00000812-R   2512    (47)    2722    (52)     2750    (52)     2933    (52)     2961    (52)
B_DATA_PKT_5                   00000817-R   2520    (47)    2726    (52)     2754    (52)     2937    (52)     2965    (52)
B_DATA_PKT_6                   0000089B-R   2528    (48)    2727    (52)     2728    (52)     2729    (52)     2764    (52)
                                                            2765    (52)     2766    (52)     2767    (52)     2775    (52)
                                                            2776    (52)     2777    (52)     2778    (52)     2887    (52)
                                                            2888    (52)     2889    (52)     2890    (52)     2904    (52)
                                                            2905    (52)     2906    (52)     2907    (52)     2938    (52)
                                                            2939    (52)     2940    (52)     2975    (52)     2976    (52)
                                                            2977    (52)     2978    (52)     2986    (52)     2987    (52)
                                                            2988    (52)     2989    (52)     3098    (52)     3099    (52)
                                                            3100    (52)     3101    (52)     3115    (52)     3116    (52)
                                                            3117    (52)     3118    (52)
B_DATA_PKT_7                   0000091F-R   2536    (48)    2768    (52)     2769    (52)     2770    (52)     2771    (52)
                                                            2779    (52)     2780    (52)     2781    (52)     2782    (52)
                                                            2979    (52)     2980    (52)     2981    (52)     2982    (52)
                                                            2990    (52)     2991    (52)     2992    (52)     2993    (52)
B_DATA_PKT_8                   000009A3-R   2544    (48)    2786    (52)     2787    (52)     2788    (52)     2789    (52)
                                                            2793    (52)     2794    (52)     2795    (52)     2796    (52)
                                                            2997    (52)     2998    (52)     2999    (52)     3000    (52)
                                                            3004    (52)     3005    (52)     3006    (52)     3007    (52)
B_DATA_PKT_9                   00000A27-R   2552    (48)    2800    (52)     2801    (52)     2802    (52)     2803    (52)
                                                            2807    (52)     2808    (52)     2809    (52)     2810    (52)
                                                            3011    (52)     3012    (52)     3013    (52)     3014    (52)
                                                            3018    (52)     3019    (52)     3020    (52)     3021    (52)
B_DIAGNOSE_CMD                 00000040-R   403     (5)     3884    (58)
B_END_PACKET_1                 000005D4-R   2135    (40)    2901    (52)     3795    (57)     3897    (58)     3999    (59)
B_END_PACKET_10                00000652-R   2287    (43)
B_END_PACKET_2                 000005E2-R   2152    (40)    4260    (61)
B_END_PACKET_3                 000005F0-R   2169    (40)    4334    (61)
B_END_PACKET_4                 000005FE-R   2186    (41)    2712    (52)     2719    (52)     2730    (52)     2737    (52)
                                                            2790    (52)     2797    (52)     2804    (52)     2811    (52)
                                                            2818    (52)     2825    (52)     2832    (52)     2839    (52)
                                                            2891    (52)     2898    (52)     2908    (52)     4498    (62)
                                                            4637    (63)     4786    (64)     4926    (65)
B_END_PACKET_5                 0000060C-R   2203    (41)    2743    (52)     2747    (52)     2755    (52)     2759    (52)
                                                            2855    (52)     2859    (52)     2863    (52)     2867    (52)
                                                            2871    (52)     2875    (52)     2879    (52)     2883    (52)
                                                            5071    (66)     5203    (67)     6466    (72)     6605    (73)
B_END_PACKET_6                 0000061A-R   2220    (41)    2723    (52)     2751    (52)
B_END_PACKET_7                 00000628-R   2237    (42)    2772    (52)     2783    (52)
B_END_PACKET_8                 00000636-R   2254    (42)
B_END_PACKET_9                 00000644-R   2270    (42)    2846    (52)     2851    (52)
B_FLAGS                        00000000-R   155     (2)     3291    (53)     3295    (53)   #-3343    (54)     3349    (54)
                                                          #-3351    (54)     3356    (54)   #-3363    (54)   #-3366    (54)
                                                          #-3377    (54)     3383    (54)   #-3385    (54)     3390    (54)
                                                          #-3397    (54)   #-3400    (54)   #-3414    (54)     3421    (54)
                                                          #-3439    (54)     3445    (54)   #-3458    (54)     3464    (54)
                                                          #-3522    (54)   #-3679    (56)     3685    (56)     3692    (56)
                                                          #-3777    (57)     3785    (57)   #-3799    (57)   #-3879    (58)
                                                            3887    (58)   #-3901    (58)   #-3981    (59)     3989    (59)
```

```
                                                    #-4003   (59)   #-4071   (60)     4076   (60)     4083   (60)
                                                    #-4233   (61)     4242   (61)     4287   (61)     4323   (61)
                                                    #-4339   (61)   #-4441   (62)     4450   (62)     4485   (62)
                                                    #-4502   (62)   #-4594   (63)     4603   (63)   #-4617   (63)
                                                    #-4730   (64)     4739   (64)     4774   (64)   #-4790   (64)
                                                    #-4883   (65)     4892   (65)   #-4906   (65)   #-5015   (66)
                                                      5026   (66)     5060   (66)   #-5075   (66)   #-5162   (67)
                                                      5171   (67)   #-5185   (67)   #-5356   (68)     5365   (68)
                                                      5400   (68)   #-5417   (68)   #-5440   (68)     5449   (68)
                                                    #-5463   (68)   #-5635   (69)     5644   (69)     5678   (69)
                                                    #-5694   (69)   #-5716   (69)     5725   (69)   #-5739   (69)
                                                    #-5902   (70)     5911   (70)     5946   (70)   #-5967   (70)
                                                    #-5984   (70)     5993   (70)   #-6010   (70)   #-6177   (71)
                                                      6186   (71)     6220   (71)   #-6239   (71)   #-6255   (71)
                                                      6264   (71)   #-6281   (71)   #-6398   (72)     6409   (72)
                                                    #-6439   (72)     6452   (72)   #-6470   (72)   #-6561   (73)
                                                      6573   (73)   #-6578   (73)     6584   (73)   #-6590   (73)
                                                    #-6677   (74)     6684   (74)   #-6690   (74)     6699   (74)
                                                      6717   (74)   #-6768   (76)     6774   (76)     6781   (76)
                                                    #-6866   (77)     6874   (77)   #-6888   (77)   #-6968   (78)
                                                      6976   (78)   #-6990   (78)   #-7070   (79)     7078   (79)
                                                    #-7092   (79)   #-7160   (80)     7165   (80)     7172   (80)
                                                    #-7322   (81)     7331   (81)     7376   (81)     7412   (81)
                                                    #-7428   (81)   #-7530   (82)     7539   (82)     7574   (82)
                                                    #-7591   (82)   #-7683   (83)     7692   (83)   #-7706   (83)
                                                    #-7819   (84)     7828   (84)     7863   (84)   #-7879   (84)
                                                    #-7972   (85)     7981   (85)   #-7995   (85)   #-8104   (86)
                                                      8115   (86)     8149   (86)   #-8164   (86)   #-8251   (87)
                                                      8260   (87)   #-8274   (87)   #-8445   (88)     8454   (88)
                                                      8489   (88)   #-8506   (88)   #-8529   (88)     8538   (88)
                                                    #-8552   (88)   #-8724   (89)     8733   (89)     8767   (89)
                                                    #-8783   (89)   #-8805   (89)     8814   (89)   #-8828   (89)
                                                    #-8991   (90)     9000   (90)     9035   (90)   #-9056   (90)
                                                    #-9073   (90)     9082   (90)   #-9099   (90)   #-9266   (91)
                                                      9275   (91)     9309   (91)   #-9328   (91)   #-9344   (91)
                                                      9353   (91)   #-9370   (91)   #-9487   (92)     9498   (92)
                                                    #-9528   (92)     9541   (92)   #-9559   (92)   #-9650   (93)
                                                      9661   (93)   #-9666   (93)     9672   (93)   #-9678   (93)
                                                    #-9765   (94)     9772   (94)   #-9778   (94)     9787   (94)
                                                      9806   (94)
B FLAGS2                        00000001-R    156     (2)   #-3680   (56)     3686   (56)     3693   (56)     3704   (56)
                                                    #-3705   (56)   #-4072   (60)     4077   (60)     4084   (60)
                                                      4095   (60)   #-4096   (60)   #-5897   (70)   #-5898   (70)
                                                      5954   (70)   #-5956   (70)     6002   (70)     6030   (70)
                                                    #-6032   (70)   #-6172   (71)   #-6173   (71)     6227   (71)
                                                    #-6228   (71)     6273   (71)     6298   (71)   #-6299   (71)
                                                    #-6440   (72)   #-6562   (73)   #-6678   (74)   #-6769   (76)
                                                      6775   (76)     6782   (76)     6793   (76)   #-6794   (76)
                                                    #-7161   (80)     7166   (80)     7173   (80)     7184   (80)
                                                    #-7185   (80)   #-8986   (90)   #-8987   (90)     9043   (90)
                                                    #-9045   (90)     9091   (90)     9119   (90)   #-9121   (90)
                                                    #-9261   (91)   #-9262   (91)     9316   (91)   #-9317   (91)
                                                      9362   (91)     9387   (91)   #-9388   (91)   #-9529   (92)
                                                    #-9651   (93)   #-9766   (94)
B_FLAGS3                        00000002-R    157     (2)   #-3562   (54)   #-3572   (54)   #-3596   (54)   #-5016   (66)
```

```
                                                              #-5066     (66)     #-6399     (72)     #-6441     (72)       6451     (72)
                                                              #-6461     (72)     #-6679     (74)       6700     (74)     #-6708     (74)
                                                                6716     (74)     #-8105     (86)     #-8155     (86)     #-9488     (92)
                                                              #-9530     (92)       9540     (92)     #-9550     (92)     #-9767     (94)
                                                                9788     (94)     #-9797     (94)       9805     (94)
B_INIT_CMD                       00000024-R    369       (5)    3782     (57)
B_INIT_DRIVE                     0000001F-R    354       (5)    3354     (54)       3388     (54)
B_NOP_CMD                        00000032-R    386       (5)    3986     (59)       6681     (74)
B_POS_1_CMD                      000002EE-R    1231      (22)   2900     (52)
B_READ_10_CMD                    00000174-R    775       (13)   2792     (52)
B_READ_11_CMD                    00000190-R    809       (13)   2806     (52)
B_READ_12_CMD                    000001AC-R    843       (14)
B_READ_13_CMD                    000001C8-R    877       (15)   2820     (52)
B_READ_14_CMD                    000001E4-R    911       (15)   2834     (52)
B_READ_15_CMD                    00000200-R    944       (16)
B_READ_16_CMD                    0000021C-R    977       (17)   2848     (52)
B_READ_17_CMD                    00000238-R    1011      (17)   2857     (52)
B_READ_18_CMD                    00000254-R    1044      (18)   2865     (52)
B_READ_19_CMD                    00000270-R    1078      (19)
B_READ_1_CMD                     0000004E-R    420       (6)    4239     (61)
B_READ_20_CMD                    0000028C-R    1112      (19)   2873     (52)
B_READ_21_CMD                    000002A8-R    1145      (20)   2881     (52)
B_READ_22_CMD                    000002C4-R    1178      (21)
B_READ_23_CMD                    000002D2-R    1195      (21)   2886     (52)       2903     (52)
B_READ_24_CMD                    000002E0-R    1214      (21)   2893     (52)
B_READ_2_CMD                     00000078-R    470       (7)    4600     (63)
B_READ_3_CMD                     00000094-R    504       (7)    5168     (67)       6570     (73)
B_READ_4_CMD                     000000B0-R    537       (8)    4889     (65)
B_READ_5_CMD                     000000E8-R    605       (9)    2725     (52)
B_READ_6_CMD                     000000F6-R    622       (10)   2732     (52)
B_READ_7_CMD                     0000012E-R    690       (11)   2753     (52)
B_READ_8_CMD                     0000013C-R    707       (11)   2757     (52)
B_READ_9_CMD                     00000158-R    741       (12)   2774     (52)
B_RESPONSE                       00000012-R    164       (2)    4280     (61)       7369     (81)
B_WRITE_10_CMD                   00000120-R    673       (11)   2749     (52)
B_WRITE_11_CMD                   0000014A-R    724       (12)   2763     (52)
B_WRITE_12_CMD                   00000166-R    758       (12)   2785     (52)
B_WRITE_13_CMD                   00000182-R    792       (13)   2799     (52)
B_WRITE_14_CMD                   0000019E-R    826       (14)
B_WRITE_15_CMD                   000001BA-R    860       (14)   2813     (52)
B_WRITE_16_CMD                   000001D6-R    894       (15)   2827     (52)
B_WRITE_17_CMD                   000001F2-R    927       (16)
B_WRITE_18_CMD                   0000020E-R    961       (16)   2843     (52)
B_WRITE_19_CMD                   0000022A-R    994       (17)   2853     (52)
B_WRITE_1_CMD                    0000005C-R    437       (6)    4284     (61)
B_WRITE_20_CMD                   00000246-R    1027      (18)   2861     (52)
B_WRITE_21_CMD                   00000262-R    1061      (18)
B_WRITE_22_CMD                   0000027E-R    1095      (19)   2869     (52)
B_WRITE_23_CMD                   0000029A-R    1128      (20)   2877     (52)
B_WRITE_24_CMD                   000002B6-R    1162      (20)
B_WRITE_2_CMD                    0000006A-R    454       (6)    4447     (62)
B_WRITE_3_CMD                    00000086-R    487       (7)    5023     (66)       6406     (72)
B_WRITE_4_CMD                    000000A2-R    520       (8)    4736     (64)
B_WRITE_5_CMD                    000000BE-R    554       (8)    2707     (52)
B_WRITE_6_CMD                    000000CC-R    571       (9)    2714     (52)
```

```
B_WRITE_7_CMD                   000000DA-R   588     (9)        2721    (52)
B_WRITE_8_CMD                   00000104-R   639     (10)       2741    (52)
B_WRITE_9_CMD                   00000112-R   656     (10)       2745    (52)
B_ZEROS                         00000021-R   358     (5)        3346    (54)      3380    (54)
CEP_FUNCTIONAL                 =00000000     72      (2)
CEP_REPAIR                     =00000001     72      (2)        72      (2)
CHECKSUM_01                    =00000001     216     (3)        2508    (47)      2517    (47)      2532    (48)      2548    (48)
                                                                2556    (48)      2605    (49)      2614    (49)      2623    (49)
                                                                2632    (50)      2640    (50)      2648    (50)
CHECKSUM_02                    =00000002     208     (3)        1279    (22)      2631    (50)      399     (5)
CHECKSUM_03                    =00000003     209     (3)        1262    (22)      2639    (50)      382     (5)
CHECKSUM_04                    =00000004     210     (3)        1208    (21)      1634    (29)      1702    (30)      2088    (38)
                                                                754     (12)      822     (13)
CHECKSUM_05                    =00000005     211     (3)        1617    (29)      1685    (30)      1736    (31)      737     (12)
                                                                805     (13)      856     (14)
CHECKSUM_08                    =00000008     212     (3)        1870    (34)      990     (17)
CHECKSUM_09                    =00000009     213     (3)        1296    (22)      1854    (33)      416     (5)       974     (16)
CHECKSUM_0A                    =0000000A     214     (3)        1245    (22)      2149    (40)      2166    (40)      2183    (40)
                                                                2217    (41)      2234    (41)      2301    (43)      383     (5)
                                                                400     (5)       417     (5)       434     (6)       451     (6)
                                                                602     (9)
CHECKSUM_0B                    =0000000B     215     (3)        2284    (42)
CHECKSUM_0C                    =0000000C     217     (3)        1209    (21)      1228    (21)      2200    (41)      2268    (42)
                                                                468     (6)       484     (7)       568     (8)       585     (9)
                                                                619     (9)       636     (10)      840     (14)      874     (14)
                                                                891     (15)      941     (16)      958     (16)
CHECKSUM_0D                    =0000000D     218     (3)        534     (8)       551     (8)       772     (12)      789     (13)
                                                                806     (13)      823     (13)      857     (14)      908     (15)
                                                                925     (15)
CHECKSUM_0E                    =0000000E     219     (3)        1363    (24)      2251    (42)      483     (7)       738     (12)
                                                                755     (12)
CHECKSUM_0F                    =0000000F     220     (3)        1347    (23)      467     (6)
CHECKSUM_10                    =00000010     221     (3)        1430    (25)      550     (8)
CHECKSUM_11                    =00000011     222     (3)        1413    (25)      2664    (50)      533     (8)
CHECKSUM_12                    =00000012     223     (3)        1263    (22)      1280    (22)      1297    (22)      1314    (23)
                                                                1331    (23)      1482    (26)      2125    (39)      2319    (43)
                                                                2336    (43)      2353    (43)      2387    (44)      2404    (44)
                                                                2471    (46)
CHECKSUM_13                    =00000013     224     (3)        1498    (26)      1566    (28)      2454    (45)      618     (9)
                                                                686     (11)
CHECKSUM_14                    =00000014     225     (3)        1348    (23)      1364    (24)      1447    (25)      1448    (25)
                                                                1465    (26)      1499    (26)      1516    (27)      1720    (31)
                                                                1754    (31)      1771    (32)      1821    (33)      1838    (33)
                                                                2089    (38)      2108    (38)      2370    (44)      2438    (45)
                                                                567     (8)
CHECKSUM_15                    =00000015     226     (3)        1414    (25)      1431    (25)      1481    (26)      1652    (29)
                                                                1669    (30)      1686    (30)      1703    (30)      1737    (31)
                                                                1788    (32)      1805    (32)      601     (9)
CHECKSUM_16                    =00000016     227     (3)        1618    (29)      1635    (29)      2421    (45)      2672    (50)
                                                                2680    (51)
CHECKSUM_19                    =00000019     228     (3)        1227    (21)      1770    (32)      1804    (32)      2107    (38)
                                                                890     (15)      924     (15)
CHECKSUM_1A                    =0000001A     229     (3)        1753    (31)      1787    (32)      873     (14)      907     (15)
CHECKSUM_1C                    =0000001C     230     (3)        1244    (22)      2124    (39)
CHECKSUM_1D                    =0000001D     231     (3)        1330    (23)      450     (6)
```

```
CHECKSUM_2D                    =0000002D     232     (3)      2687    (51)      2696    (51)
CHECKSUM_36                    =00000036     233     (3)      2267    (42)      2437    (45)
CHECKSUM_41                    =00000041     234     (3)      2540    (48)      2564    (48)      2589    (49)      2597    (49)
CHECKSUM_42                    =00000042     235     (3)      2148    (40)      2199    (41)      2250    (42)      2283    (42)
                                                              2318    (43)      2369    (44)      2420    (45)      2453    (45)
CHECKSUM_43                    =00000043     236     (3)      2233    (41)      2403    (44)
CHECKSUM_54                    =00000054     237     (3)
CHECKSUM_5C                    =0000005C     238     (3)      2182    (40)      2352    (43)
CHECKSUM_6E                    =0000006E     239     (3)      1074    (18)      1954    (35)
CHECKSUM_6F                    =0000006F     240     (3)      1175    (20)      2055    (37)
CHECKSUM_72                    =00000072     241     (3)      2656    (50)
CHECKSUM_75                    =00000075     242     (3)      2572    (48)      2581    (49)
CHECKSUM_79                    =00000079     243     (3)      1820    (33)      940     (16)
CHECKSUM_80                    =00000080     244     (3)      2509    (47)      2526    (47)      2533    (48)      2541    (48)
CHECKSUM_82                    =00000082     245     (3)      2671    (50)
CHECKSUM_83                    =00000083     246     (3)      2647    (50)      2679    (51)
CHECKSUM_84                    =00000084     247     (3)      1668    (30)      1837    (33)      788     (13)      957     (16)
CHECKSUM_85                    =00000085     248     (3)      1651    (29)      2590    (49)      771     (12)
CHECKSUM_88                    =00000088     249     (3)      1024    (17)      1904    (34)
CHECKSUM_89                    =00000089     250     (3)      1007    (17)      1091    (19)      1887    (34)      1971    (36)
CHECKSUM_8A                    =0000008A     251     (3)      1057    (18)      1191    (21)      1313    (23)      1937    (35)
                                                              2071    (38)      433     (6)       501     (7)       518     (7)
                                                              653     (10)      687     (11)      704     (11)
CHECKSUM_8B                    =0000008B     252     (3)      1040    (18)      1920    (35)
CHECKSUM_8C                    =0000008C     253     (3)      1109    (19)      1126    (19)      1176    (20)      1192    (21)
                                                              670     (10)      721     (11)
CHECKSUM_8D                    =0000008D     255     (4)      975     (16)      991     (17)
CHECKSUM_90                    =00000090     256     (4)      1008    (17)      1025    (17)      1041    (18)      1058    (18)
                                                              1075    (18)      1092    (19)      1142    (20)      1159    (20)
CHECKSUM_91                    =00000091     257     (4)      1583    (28)      1600    (28)      703     (11)      720     (11)
CHECKSUM_92                    =00000092     258     (4)      1381    (24)      1397    (24)      1398    (24)      1532    (27)
                                                              1533    (27)      1549    (27)      1567    (28)      1584    (28)
                                                              517     (7)       652     (10)      669     (10)
CHECKSUM_93                    =00000093     259     (4)      1380    (24)      1515    (27)      500     (7)       635     (10)
CHECKSUM_94                    =00000094     260     (4)      1464    (26)      1550    (27)      1601    (28)      1989    (36)
                                                              2006    (36)      2056    (37)      2072    (38)      584     (9)
CHECKSUM_95                    =00000095     261     (4)      1855    (33)      1871    (34)
CHECKSUM_98                    =00000098     262     (4)      1888    (34)      1905    (34)      1921    (35)      1938    (35)
                                                              1955    (35)      1972    (36)      2022    (37)      2039    (37)
CHECKSUM_A6                    =000000A6     263     (4)      2300    (43)      2470    (46)
CHECKSUM_A9                    =000000A9     264     (4)
CHECKSUM_AA                    =000000AA     265     (4)
CHECKSUM_AB                    =000000AB     266     (4)      2516    (47)      2525    (47)
CHECKSUM_AE                    =000000AE     267     (4)      2606    (49)      2615    (49)
CHECKSUM_BA                    =000000BA     268     (4)      2493    (47)      2624    (49)
CHECKSUM_BF                    =000000BF     269     (4)      2492    (47)      2501    (47)
CHECKSUM_C0                    =000000C0     270     (4)      2557    (48)      2565    (48)
CHECKSUM_C1                    =000000C1     271     (4)      2500    (47)
CHECKSUM_C2                    =000000C2     272     (4)      2165    (40)      2216    (41)      2335    (43)      2386    (44)
CHECKSUM_C5                    =000000C5     273     (4)      2598    (49)
CHECKSUM_C9                    =000000C9     274     (4)      2688    (51)
CHECKSUM_DE                    =000000DE     275     (4)      1125    (19)      1158    (20)      2005    (36)      2038    (37)
CHECKSUM_DF                    =000000DF     276     (4)      1108    (19)      1141    (20)      1988    (36)      2021    (37)
CHECKSUM_E0                    =000000E0     277     (4)      2549    (48)
CHECKSUM_E5                    =000000E5     278     (4)      2697    (51)
```

```
CHECKSUM_E8                   =000000E8   279    (4)     2573    (48)
CHECKSUM_F4                   =000000F4   280    (4)     2582    (49)
CHECKSUM_F7                   =000000F7   281    (4)     2663    (50)
CHECKSUM_FA                   =000000FA   282    (4)     1719    (31)   839     (14)
CHECKSUM_FC                   =000000FC   283    (4)     2655    (50)
CHECK_PACKET                   0000213B-R 3504   (54)    3702    (56)   3808    (57)   3910    (58)   4012    (59)
                                                         4093    (60)   4265    (61)   4277    (61)   4312    (61)
                                                         4349    (61)   4474    (62)   4512    (62)   4628    (63)
                                                         4640    (63)   4763    (64)   4800    (64)   4917    (65)
                                                         4929    (65)   5049    (66)   5085    (66)   5195    (67)
                                                         5206    (67)   5389    (68)   5427    (68)   5474    (68)
                                                         5486    (68)   5667    (69)   5704    (69)   5749    (69)
                                                         5760    (69)   5935    (70)   5977    (70)   6021    (70)
                                                         6037    (70)   6209    (71)   6249    (71)   6291    (71)
                                                         6306    (71)   6432    (72)   6480    (72)   6599    (73)
                                                         6609    (73)   6791    (76)   6897    (77)   6999    (78)
                                                         7101    (79)   7182    (80)   7354    (81)   7366    (81)
                                                         7401    (81)   7438    (81)   7563    (82)   7601    (82)
                                                         7717    (83)   7729    (83)   7852    (84)   7889    (84)
                                                         8006    (85)   8018    (85)   8138    (86)   8174    (86)
                                                         8284    (87)   8295    (87)   8478    (88)   8516    (88)
                                                         8563    (88)   8575    (88)   8756    (89)   8793    (89)
                                                         8838    (89)   8849    (89)   9024    (90)   9066    (90)
                                                         9110    (90)   9126    (90)   9298    (91)   9338    (91)
                                                         9380    (91)   9395    (91)   9521    (92)   9569    (92)
                                                         9687    (93)   9697    (93)
CLEAR_BUFFER                   00002134-R 3487   (54)  #-3490    (54)   3678    (56)   3780    (57)   3882    (58)
                                                         3984    (59)   4069    (60)   4236    (61)   4444    (62)
                                                         4597    (63)   4733    (64)   4886    (65)   5020    (66)
                                                         5165    (67)   5359    (68)   5443    (68)   5638    (69)
                                                         5719    (69)   5905    (70)   5987    (70)   6180    (71)
                                                         6258    (71)   6403    (72)   6565    (73)   6767    (76)
                                                         6869    (77)   6971    (78)   7073    (79)   7158    (80)
                                                         7325    (81)   7533    (82)   7686    (83)   7822    (84)
                                                         7975    (85)   8109    (86)   8254    (87)   8448    (88)
                                                         8532    (88)   8727    (89)   8808    (89)   8994    (90)
                                                         9076    (90)   9269    (91)   9347    (91)   9492    (92)
                                                         9654    (93)
COMMAND FLAG                  =00000002   103    (2)     1012    (17)   1028    (18)   1045    (18)   1062    (18)
                                                         1079    (19)   1096    (19)   1113    (19)   1129    (20)
                                                         1146    (20)   1163    (20)   1179    (21)   1196    (21)
                                                         1215    (21)   1232    (22)   1250    (22)   1267    (22)
                                                         1284    (22)   1301    (23)   1318    (23)   1335    (23)
                                                         1351    (24)   1368    (24)   1385    (24)   1401    (25)
                                                         1418    (25)   1435    (25)   1452    (26)   1469    (26)
                                                         1486    (26)   1503    (27)   1520    (27)   1537    (27)
                                                         1554    (28)   1571    (28)   1588    (28)   1605    (29)
                                                         1622    (29)   1639    (29)   1656    (30)   1673    (30)
                                                         1690    (30)   1707    (31)   1724    (31)   1741    (31)
                                                         1758    (32)   1775    (32)   1792    (32)   1808    (33)
                                                         1825    (33)   1842    (33)   1858    (34)   1875    (34)
                                                         1892    (34)   1908    (35)   1925    (35)   1942    (35)
                                                         1959    (36)   1976    (36)   1993    (36)   2009    (37)
                                                         2026    (37)   2043    (37)   2059    (38)   2076    (38)
                                                         2095    (38)   2112    (39)   370     (5)    387     (5)
```

```
                                              404   (5)    421   (6)    438   (6)    455   (6)
                                              471   (7)    488   (7)    505   (7)    521   (8)
                                              538   (8)    555   (8)    572   (9)    589   (9)
                                              606   (9)    623   (10)   640   (10)   657   (10)
                                              674   (11)   691   (11)   708   (11)   725   (12)
                                              742   (12)   759   (12)   776   (13)   793   (13)
                                              810   (13)   827   (14)   844   (14)   861   (14)
                                              878   (15)   895   (15)   912   (15)   928   (16)
                                              945   (16)   962   (16)   978   (17)   995   (17)
COMMAND_TBL_1        0000132B-R  2707  (52)   5354  (68)
COMMAND_TBL_2        00001397-R  2741  (52)   5633  (69)
COMMAND_TBL_3        000013D3-R  2763  (52)   5900  (70)
COMMAND_TBL_4        000014E3-R  2843  (52)   6175  (71)
COMMAND_TBL_5        00001563-R  2886  (52)
CONTINUE_FLAG       =00000010    106   (2)    362   (5)
DATA_0000           =00000000    319   (4)    2480  (47)   2482  (47)   2488  (47)   2489  (47)
                                              2490  (47)   2524  (47)   2531  (48)   2580  (49)
                                              2613  (49)   2622  (49)   2695  (51)   359   (5)
DATA_0001           =00000001    320   (4)    2481  (47)   2539  (48)
DATA_0004           =00000004    321   (4)    2479  (47)
DATA_000F           =0000000F    322   (4)    2483  (47)
DATA_0015           =00000015    323   (4)    2588  (49)
DATA_0080           =00000080    324   (4)    2604  (49)   2612  (49)   2621  (49)
DATA_009             00000014-R  3668  (56)   3668  (56)
DATA_00AA           =000000AA    325   (4)    2515  (47)   2523  (47)
DATA_00FF           =000000FF    326   (4)    2499  (47)
DATA_010             0000001C-R  4059  (60)   4059  (60)
DATA_0100           =00000100    327   (4)    2555  (48)
DATA_0101           =00000101    328   (4)    2563  (48)   2571  (48)   2579  (49)
DATA_011             00000020-R  6757  (76)   6757  (76)
DATA_0115           =00000115    329   (4)    2596  (49)
DATA_012             00000028-R  7148  (80)   7148  (80)
DATA_0180           =00000180    330   (4)    2547  (48)
DATA_0204           =00000204    331   (4)    2630  (50)
DATA_0206           =00000206    332   (4)    2686  (51)   2694  (51)
DATA_025A           =0000025A    333   (4)    2670  (50)
DATA_0400           =00000400    335   (4)    2484  (47)
DATA_0604           =00000604    336   (4)    2638  (50)
DATA_0605           =00000605    337   (4)    2654  (50)   2662  (50)
DATA_0606           =00000606    338   (4)    2646  (50)
DATA_065A           =0000065A    334   (4)    2678  (51)
DATA_0800           =00000800    339   (4)    2491  (47)
DATA_0960           =00000960    340   (4)    2486  (47)
DATA_1400           =00001400    341   (4)    2487  (47)
DATA_1D14           =00001D14    342   (4)
DATA_4A77           =00004A77    343   (4)
DATA_774A           =0000774A    345   (4)    2485  (47)
DATA_7D09           =00007D09    344   (4)
DATA_ERR_CONT        00001E4D-R  3260  (53)   3708  (56)   4099  (60)   4476  (62)   4765  (64)
                                              5051  (66)   5391  (68)   5669  (69)   5937  (70)
                                              6211  (71)   6434  (72)   6797  (76)   7188  (80)
                                              7565  (82)   7854  (84)   8140  (86)   8480  (88)
                                              8758  (89)   9026  (90)   9300  (91)   9523  (92)
DATA_ERR_DATA        00001E75-R  3267  (53)   4314  (61)   4630  (63)   4919  (65)   5197  (67)
                                              5476  (68)   5751  (69)   6023  (70)   6293  (71)
```

```
                                                               6601    (73)      7403    (81)      7719    (83)      8008    (85)
                                                               8286    (87)      8565    (88)      8840    (89)      9112    (90)
                                                               9382    (91)      9689    (93)
DATA_ERR_END                     00001E9D-R   3274    (53)     3810    (57)      3912    (58)      4014    (59)      4267    (61)
                                                               4351    (61)      4514    (62)      4642    (63)      4802    (64)
                                                               4931    (65)      5087    (66)      5208    (67)      5429    (68)
                                                               5488    (68)      5706    (69)      5762    (69)      5979    (70)
                                                               6039    (70)      6251    (71)      6308    (71)      6482    (72)
                                                               6611    (73)      6899    (77)      7001    (78)      7103    (79)
                                                               7356    (81)      7440    (81)      7603    (82)      7731    (83)
                                                               7891    (84)      8020    (85)      8176    (86)      8297    (87)
                                                               8518    (88)      8577    (88)      8795    (89)      8851    (89)
                                                               9068    (90)      9128    (90)      9340    (91)      9397    (91)
                                                               9571    (92)      9699    (93)
DATA_FFFF                       =0000FFFF     346     (4)      2507    (47)
DATA_FLAG                       =00000001     102     (2)      2477    (47)      2497    (47)      2505    (47)      2513    (47)
                                                               2521    (47)      2529    (48)      2537    (48)      2545    (48)
                                                               2553    (48)      2561    (48)      2569    (48)      2577    (49)
                                                               2586    (49)      2594    (49)      2602    (49)      2610    (49)
                                                               2619    (49)      2628    (50)      2636    (50)      2644    (50)
                                                               2652    (50)      2660    (50)      2668    (50)      2676    (51)
                                                               2684    (51)      2692    (51)
DEFAULT                         =00000000     94      (2)      3668    (56)      4059    (60)      6757    (76)      7148    (80)
DELAY_TIME1                     =FFFB6C20     196     (2)     #-3357   (54)     #-3391   (54)     #-3425   (54)     #-3470   (54)
                                                              #-3781   (57)     #-3791   (57)     #-3883   (58)     #-3985   (59)
                                                              #-3995   (59)     #-4319   (61)     #-4481   (62)     #-4770   (64)
                                                              #-5056   (66)     #-5396   (68)     #-5445   (68)     #-5674   (69)
                                                              #-5721   (69)     #-5942   (70)     #-5989   (70)     #-6216   (71)
                                                              #-6260   (71)     #-6680   (74)     #-6870   (77)     #-6880   (77)
                                                              #-6972   (78)     #-7074   (79)     #-7084   (79)     #-7408   (81)
                                                              #-7570   (82)     #-7859   (84)     #-8145   (86)     #-8485   (88)
                                                              #-8534   (88)     #-8763   (89)     #-8810   (89)     #-9031   (90)
                                                              #-9078   (90)     #-9305   (91)     #-9349   (91)     #-9768   (94)
DELAY_TIME2                     =FFF85EE0     197     (2)     #-3344   (54)     #-3352   (54)     #-3378   (54)     #-3386   (54)
DELAY_TIME3                     =FFF3CB00     198     (2)     #-3893   (58)     #-4293   (61)     #-4446   (62)     #-4494   (62)
                                                              #-4599   (63)     #-4609   (63)     #-4735   (64)     #-4746   (64)
                                                              #-4888   (65)     #-4898   (65)     #-5032   (66)     #-5167   (67)
                                                              #-5177   (67)     #-5361   (68)     #-5640   (69)     #-5650   (69)
                                                              #-5731   (69)     #-5907   (70)     #-5999   (70)     #-6182   (71)
                                                              #-6270   (71)     #-6415   (72)     #-6693   (74)     #-6982   (78)
                                                              #-7382   (81)     #-7535   (82)     #-7583   (82)     #-7688   (83)
                                                              #-7698   (83)     #-7824   (84)     #-7835   (84)     #-7977   (85)
                                                              #-7987   (85)     #-8121   (86)     #-8256   (87)     #-8266   (87)
                                                              #-8450   (88)     #-8729   (89)     #-8739   (89)     #-8820   (89)
                                                              #-8996   (90)     #-9088   (90)     #-9271   (91)     #-9359   (91)
                                                              #-9504   (92)     #-9781   (94)
DELAY_TIME4                     =FFF0BDC0     199     (2)     #-3449   (54)     #-3476   (54)
DELAY_TIME5                     =FFE17B80     200     (2)     #-4782   (64)     #-5022   (66)     #-5409   (68)     #-5455   (68)
                                                              #-5959   (70)     #-6405   (72)     #-7871   (84)     #-8111   (86)
                                                              #-8498   (88)     #-8544   (88)     #-9048   (90)     #-9494   (92)
DELAY_TIME6                     =FFC2F700     201     (2)     #-4457   (62)     #-5067   (66)     #-5372   (68)     #-5686   (69)
                                                              #-7546   (82)     #-8156   (86)     #-8461   (88)     #-8775   (89)
DELAY_TIME7                     =FF7E4CE0     202     (2)     #-4238   (61)     #-4283   (61)     #-4329   (61)     #-5918   (70)
                                                              #-6192   (71)     #-6231   (71)     #-6444   (72)     #-6462   (72)
                                                              #-6579   (73)     #-6709   (74)     #-7327   (81)     #-7372   (81)
```

```
                                                          #-7418    (81)   #-9007    (90)   #-9281    (91)   #-9320    (91)
                                                          #-9533    (92)   #-9551    (92)   #-9667    (93)   #-9798    (94)
DELAY_TIME8                       =FAEF00C0    203   (2)  #-4248    (61)   #-7337    (81)
DIAGNOSE                          =00000007    117   (2)    1286    (22)      406    (5)
DS$ABORT                           00010020                 3690    (56)     3698    (56)     3711    (56)     3789    (57)
                                                            3804    (57)     3812    (57)     3891    (58)     3906    (58)
                                                            3914    (58)     3993    (59)     4008    (59)     4016    (59)
                                                            4081    (60)     4089    (60)     4102    (60)     4246    (61)
                                                            4258    (61)     4269    (61)     4291    (61)     4308    (61)
                                                            4317    (61)     4327    (61)     4345    (61)     4353    (61)
                                                            4454    (62)     4470    (62)     4479    (62)     4489    (62)
                                                            4508    (62)     4517    (62)     4607    (63)     4623    (63)
                                                            4633    (63)     4645    (63)     4743    (64)     4759    (64)
                                                            4768    (64)     4778    (64)     4796    (64)     4805    (64)
                                                            4896    (65)     4912    (65)     4922    (65)     4934    (65)
                                                            5030    (66)     5045    (66)     5054    (66)     5064    (66)
                                                            5081    (66)     5090    (66)     5175    (67)     5191    (67)
                                                            5200    (67)     5211    (67)     5369    (68)     5385    (68)
                                                            5394    (68)     5404    (68)     5423    (68)     5432    (68)
                                                            5453    (68)     5469    (68)     5479    (68)     5491    (68)
                                                            5648    (69)     5663    (69)     5672    (69)     5682    (69)
                                                            5700    (69)     5709    (69)     5729    (69)     5745    (69)
                                                            5754    (69)     5765    (69)     5915    (70)     5931    (70)
                                                            5940    (70)     5950    (70)     5973    (70)     5982    (70)
                                                            5997    (70)     6016    (70)     6026    (70)     6042    (70)
                                                            6190    (71)     6205    (71)     6214    (71)     6224    (71)
                                                            6245    (71)     6253    (71)     6268    (71)     6287    (71)
                                                            6296    (71)     6311    (71)     6413    (72)     6428    (72)
                                                            6437    (72)     6459    (72)     6476    (72)     6485    (72)
                                                            6577    (73)     6596    (73)     6604    (73)     6614    (73)
                                                            6688    (74)     6705    (74)     6722    (74)     6779    (76)
                                                            6787    (76)     6800    (76)     6878    (77)     6893    (77)
                                                            6901    (77)     6980    (78)     6995    (78)     7003    (78)
                                                            7082    (79)     7097    (79)     7105    (79)     7170    (80)
                                                            7178    (80)     7191    (80)     7335    (81)     7347    (81)
                                                            7358    (81)     7380    (81)     7397    (81)     7406    (81)
                                                            7416    (81)     7434    (81)     7442    (81)     7543    (82)
                                                            7559    (82)     7568    (82)     7578    (82)     7597    (82)
                                                            7606    (82)     7696    (83)     7712    (83)     7722    (83)
                                                            7734    (83)     7832    (84)     7848    (84)     7857    (84)
                                                            7867    (84)     7885    (84)     7894    (84)     7985    (85)
                                                            8001    (85)     8011    (85)     8023    (85)     8119    (86)
                                                            8134    (86)     8143    (86)     8153    (86)     8170    (86)
                                                            8179    (86)     8264    (87)     8280    (87)     8289    (87)
                                                            8300    (87)     8458    (88)     8474    (88)     8483    (88)
                                                            8493    (88)     8512    (88)     8521    (88)     8542    (88)
                                                            8558    (88)     8568    (88)     8580    (88)     8737    (89)
                                                            8752    (89)     8761    (89)     8771    (89)     8789    (89)
                                                            8798    (89)     8818    (89)     8834    (89)     8843    (89)
                                                            8854    (89)     9004    (90)     9020    (90)     9029    (90)
                                                            9039    (90)     9062    (90)     9071    (90)     9086    (90)
                                                            9105    (90)     9115    (90)     9131    (90)     9279    (91)
                                                            9294    (91)     9303    (91)     9313    (91)     9334    (91)
                                                            9342    (91)     9357    (91)     9376    (91)     9385    (91)
                                                            9400    (91)     9502    (92)     9517    (92)     9526    (92)
```

```
                                         9548   (92)     9565   (92)     9574   (92)     9665   (93)
                                         9684   (93)     9692   (93)     9702   (93)     9776   (94)
                                         9793   (94)     9811   (94)
DS$ASKLGCL             00010098          4280   (61)     7369   (81)
DS$BGNSUB              00010030          3775   (57)     3876   (58)     3978   (59)     4231   (61)
                                         4439   (62)     4592   (63)     4728   (64)     4881   (65)
                                         5014   (66)     5160   (67)     5350   (68)     5629   (69)
                                         5896   (70)     6171   (71)     6396   (72)     6559   (73)
                                         6674   (74)     6864   (77)     6965   (78)     7067   (79)
                                         7320   (81)     7528   (82)     7681   (83)     7817   (84)
                                         7970   (85)     8103   (86)     8249   (87)     8439   (88)
                                         8718   (89)     8985   (90)     9260   (91)     9485   (92)
                                         9648   (93)     9762   (94)
DS$BREAK               00010058          3701   (56)     3807   (57)     3909   (58)     4011   (59)
                                         4022   (59)     4092   (60)     4264   (61)     4276   (61)
                                         4311   (61)     4348   (61)     4473   (62)     4511   (62)
                                         4627   (63)     4639   (63)     4762   (64)     4799   (64)
                                         4916   (65)     4928   (65)     5048   (66)     5084   (66)
                                         5194   (67)     5205   (67)     5388   (68)     5426   (68)
                                         5473   (68)     5485   (68)     5666   (69)     5703   (69)
                                         5748   (69)     5759   (69)     5934   (70)     5976   (70)
                                         6020   (70)     6036   (70)     6208   (71)     6248   (71)
                                         6290   (71)     6305   (71)     6431   (72)     6450   (72)
                                         6479   (72)     6583   (73)     6598   (73)     6608   (73)
                                         6698   (74)     6715   (74)     6733   (74)     6790   (76)
                                         6896   (77)     6998   (78)     7100   (79)     7111   (79)
                                         7181   (80)     7353   (81)     7365   (81)     7400   (81)
                                         7437   (81)     7562   (82)     7600   (82)     7716   (83)
                                         7728   (83)     7851   (84)     7888   (84)     8005   (85)
                                         8017   (85)     8137   (86)     8173   (86)     8283   (87)
                                         8294   (87)     8477   (88)     8515   (88)     8562   (88)
                                         8574   (88)     8755   (89)     8792   (89)     8837   (89)
                                         8848   (89)     9023   (90)     9065   (90)     9109   (90)
                                         9125   (90)     9297   (91)     9337   (91)     9379   (91)
                                         9394   (91)     9520   (92)     9539   (92)     9568   (92)
                                         9671   (93)     9686   (93)     9696   (93)     9786   (94)
                                         9804   (94)     9822   (94)
DS$CLRVEC              00010168          4020   (59)     6675   (74)     6729   (74)     6730   (74)
                                         6731   (74)     7109   (79)     9763   (94)     9818   (94)
                                         9819   (94)     9820   (94)
DS$ENDSUB              00010038          3815   (57)     3917   (58)     4019   (59)     4358   (61)
                                         4519   (62)     4648   (63)     4807   (64)     4935   (65)
                                         5092   (66)     5213   (67)     5497   (68)     5771   (69)
                                         6046   (70)     6315   (71)     6487   (72)     6616   (73)
                                         6727   (74)     6904   (77)     7006   (78)     7108   (79)
                                         7447   (81)     7608   (82)     7737   (83)     7896   (84)
                                         8024   (85)     8181   (86)     8302   (87)     8586   (88)
                                         8860   (89)     9135   (90)     9404   (91)     9576   (92)
                                         9704   (93)     9816   (94)
DS$ERRHARD             000100D0          3687   (56)     3695   (56)     3708   (56)     3786   (57)
                                         3801   (57)     3810   (57)     3888   (58)     3903   (58)
                                         3912   (58)     3990   (59)     4005   (59)     4014   (59)
                                         4078   (60)     4086   (60)     4099   (60)     4243   (61)
                                         4254   (61)     4267   (61)     4288   (61)     4304   (61)
                                         4314   (61)     4324   (61)     4341   (61)     4351   (61)
```

```
                                                            4451     (62)         4466     (62)         4476     (62)         4486     (62)
                                                            4504     (62)         4514     (62)         4604     (63)         4619     (63)
                                                            4630     (63)         4642     (63)         4740     (64)         4755     (64)
                                                            4765     (64)         4775     (64)         4792     (64)         4802     (64)
                                                            4893     (65)         4908     (65)         4919     (65)         4931     (65)
                                                            5027     (66)         5041     (66)         5051     (66)         5061     (66)
                                                            5077     (66)         5087     (66)         5172     (67)         5187     (67)
                                                            5197     (67)         5208     (67)         5366     (68)         5381     (68)
                                                            5391     (68)         5401     (68)         5419     (68)         5429     (68)
                                                            5450     (68)         5465     (68)         5476     (68)         5488     (68)
                                                            5645     (69)         5659     (69)         5669     (69)         5679     (69)
                                                            5696     (69)         5706     (69)         5726     (69)         5741     (69)
                                                            5751     (69)         5762     (69)         5912     (70)         5927     (70)
                                                            5937     (70)         5947     (70)         5969     (70)         5979     (70)
                                                            5994     (70)         6012     (70)         6023     (70)         6039     (70)
                                                            6187     (71)         6201     (71)         6211     (71)         6221     (71)
                                                            6241     (71)         6251     (71)         6265     (71)         6283     (71)
                                                            6293     (71)         6308     (71)         6410     (72)         6424     (72)
                                                            6434     (72)         6456     (72)         6472     (72)         6482     (72)
                                                            6574     (73)         6593     (73)         6601     (73)         6611     (73)
                                                            6685     (74)         6702     (74)         6719     (74)         6776     (76)
                                                            6784     (76)         6797     (76)         6875     (77)         6890     (77)
                                                            6899     (77)         6977     (78)         6992     (78)         7001     (78)
                                                            7079     (79)         7094     (79)         7103     (79)         7167     (80)
                                                            7175     (80)         7188     (80)         7332     (81)         7343     (81)
                                                            7356     (81)         7377     (81)         7393     (81)         7403     (81)
                                                            7413     (81)         7430     (81)         7440     (81)         7540     (82)
                                                            7555     (82)         7565     (82)         7575     (82)         7593     (82)
                                                            7603     (82)         7693     (83)         7708     (83)         7719     (83)
                                                            7731     (83)         7829     (84)         7844     (84)         7854     (84)
                                                            7864     (84)         7881     (84)         7891     (84)         7982     (85)
                                                            7997     (85)         8008     (85)         8020     (85)         8116     (86)
                                                            8130     (86)         8140     (86)         8150     (86)         8166     (86)
                                                            8176     (86)         8261     (87)         8276     (87)         8286     (87)
                                                            8297     (87)         8455     (88)         8470     (88)         8480     (88)
                                                            8490     (88)         8508     (88)         8518     (88)         8539     (88)
                                                            8554     (88)         8565     (88)         8577     (88)         8734     (89)
                                                            8748     (89)         8758     (89)         8768     (89)         8785     (89)
                                                            8795     (89)         8815     (89)         8830     (89)         8840     (89)
                                                            8851     (89)         9001     (90)         9016     (90)         9026     (90)
                                                            9036     (90)         9058     (90)         9068     (90)         9083     (90)
                                                            9101     (90)         9112     (90)         9128     (90)         9276     (91)
                                                            9290     (91)         9300     (91)         9310     (91)         9330     (91)
                                                            9340     (91)         9354     (91)         9372     (91)         9382     (91)
                                                            9397     (91)         9499     (92)         9513     (92)         9523     (92)
                                                            9545     (92)         9561     (92)         9571     (92)         9662     (93)
                                                            9681     (93)         9689     (93)         9699     (93)         9773     (94)
                                                            9790     (94)         9808     (94)
DS$K_ERROR                        =00000002     90       (2)
DS$K_NORMAL                       =00000001     90       (2)
DS$K_SEVERE                       =00000004     90       (2)
DS$K_SUBSYS                       =00000066     90       (2)      90       (2)
DS$K_WARNING                      =00000000     90       (2)
DS$PRINTF                          000100F0                         3173     (53)         3178     (53)         3183     (53)
DS$PRINTX                          000100E8                         3200     (53)         3262     (53)         3263     (53)         3264     (53)
```

```
                                                             3269   (53)     3270   (53)     3271   (53)     3276   (53)
                                                             3290   (53)     3292   (53)     3294   (53)     3300   (53)
                                                             3306   (53)     3309   (53)     3310   (53)     3316   (53)
                                                             3321   (53)     3610   (54)     3614   (54)     3616   (54)
                                                             3626   (54)
DS$SETIPL                       00010178                     6692   (74)     6714   (74)     6725   (74)     9780   (94)
                                                             9803   (94)     9814   (94)
DS$SETVEC                       00010160                     3674   (56)     4064   (60)     6397   (72)     6560   (73)
                                                             6676   (74)     6763   (76)     7153   (80)     9486   (92)
                                                             9649   (93)     9764   (94)
DS$_ARITH                      =006600D0     90     (2)
DS$_ASBE                       =00660118     90     (2)
DS$_BADLINK                    =006600F0     90     (2)
DS$_BADTYPE                    =006600E8     90     (2)
DS$_BIIC                       =00660120     90     (2)
DS$_CHME                       =006600A8     90     (2)
DS$_CHMK                       =006600E0     90     (2)
DS$_DEVNAME                    =00660108     90     (2)
DS$_ERROR                      =00660002     90     (2)
DS$_FHWE                       =00660068     90     (2)
DS$_FRAGBUF                    =00660080     90     (2)
DS$_ICBUSY                     =006600C8     90     (2)
DS$_ICERR                      =006600C0     90     (2)
DS$_IHWE                       =00660060     90     (2)
DS$_ILLCHAR                    =00660018     90     (2)
DS$_ILLPAGCNT                  =00660078     90     (2)
DS$_ILLUNIT                    =00660100     90     (2)
DS$_INSFMEM                    =00660050     90     (2)
DS$_IPL2HI                     =006600B8     90     (2)
DS$_IVADDR                     =00660040     90     (2)
DS$_IVVECT                     =00660038     90     (2)
DS$_KRNLSTK                    =00660090     90     (2)
DS$_LOGIC                      =00660070     90     (2)
DS$_MCHK                       =00660088     90     (2)
DS$_MMOFF                      =00660058     90     (2)
DS$_NEEDUNIT                   =006600F8     90     (2)
DS$_NODE                       =00660128     90     (2)
DS$_NOPCS                      =00660110     90     (2)
DS$_NORMAL                     =00660001     90     (2)
DS$_NOSUPPORT                  =006600B1     90     (2)
DS$_NOTDON                     =00660030     90     (2)
DS$_NOTIMP                     =006600B0     90     (2)      90     (2)
DS$_NULLSTR                    =00660010     90     (2)
DS$_OVERFLOW                   =00660008     90     (2)
DS$_POWER                      =00660098     90     (2)
DS$_PROGERR                    =00660020     90     (2)
DS$_SEVERE                     =00660004     90     (2)
DS$_TRANSL                     =006600A0     90     (2)
DS$_TRUNCATE                   =00660028     90     (2)
DS$_UNEXPINT                   =006600D8     90     (2)
DS$_VASFULL                    =00660048     90     (2)
DS$_WARNING                    =00660000     90     (2)      90     (2)
DUMP_14                         00002227-R   3624   (54)     3297   (53)     3307   (53)
DUMP_PKT                        000021E7-R   3608   (54)     3278   (53)
END_CODE                       =00000040     118    (2)      2138   (40)     2155   (40)     2172   (40)     2189   (41)
```

```
                                                              2206    (41)      2223    (41)      2240    (42)      2257    (42)
                                                              2273    (42)      2290    (43)      2308    (43)      2325    (43)
                                                              2342    (43)      2359    (44)      2376    (44)      2393    (44)
                                                              2410    (45)      2427    (45)      2443    (45)      2460    (46)
END_FLAG                         =00000002     104      (2)   2136    (40)      2153    (40)      2170    (40)      2187    (41)
                                                              2204    (41)      2221    (41)      2238    (42)      2255    (42)
                                                              2271    (42)      2288    (43)      2306    (43)      2323    (43)
                                                              2340    (43)      2357    (44)      2374    (44)      2391    (44)
                                                              2408    (45)      2425    (45)      2441    (45)      2458    (46)
ENV$M_DOMAIN                     =00000002     72       (2)
ENV$M_LEVEL                      =00000001     72       (2)
ENV$M_SUPER                      =000003FC     72       (2)
ENV$S_DOMAIN                     =00000001     72       (2)
ENV$S_LEVEL                      =00000001     72       (2)
ENV$S_SUPER                      =00000008     72       (2)
ENV$V_DOMAIN                     =00000001     72       (2)   72      (2)
ENV$V_LEVEL                      =00000000     72       (2)   72      (2)
ENV$V_SUPER                      =00000002     72       (2)
ENV$_CPU                         =00000000     72       (2)   72      (2)
ENV$_FUNCTIONAL                  =00000000     72       (2)   72      (2)
ENV$_REPAIR                      =00000001     72       (2)   72      (2)
ENV$_SUPER                       =00000001     72       (2)
ENV$_SYSTEM                      =00000001     72       (2)   72      (2)
ERR$_MSGADR                      =0000000C     89       (2)
ERR$_NARGS                       =0000000A     89       (2)
ERR$_NUM                         =00000004     89       (2)
ERR$_P1                          =00000014     89       (2)   3292    (53)      3294    (53)
ERR$_P2                          =00000018     89       (2)
ERR$_P3                          =0000001C     89       (2)
ERR$_P4                          =00000020     89       (2)
ERR$_P5                          =00000024     89       (2)
ERR$_P6                          =00000028     89       (2)
ERR$_POINTER                     =00000010     89       (2)
ERR$_UNIT                        =00000008     89       (2)
FIND_END                          0000223D-R   3636     (54)  3304    (53)
HALT                             =00000004     94       (2)
INITIALIZE                       =00000001     113      (2)   1252    (22)      372     (5)
INIT_FLAG                        =00000004     105      (2)   355     (5)       356     (5)
INIT_SEQUENCE                     00001F59-R   3342     (54)  3582    (54)      3682    (56)      4073    (60)
INT_ERR                           00001F3D-R   3314     (53)  6702    (74)      9790    (94)
IPR                              =00000006     94       (2)
L_COUNT                           00000006-R   161      (2)  #-4625   (63)     #-4635   (63)     #-4914   (65)     #-4924   (65)
                                                             #-6018   (70)     #-6029   (70)     #-6031   (70)     #-7714   (83)
                                                             #-7724   (83)     #-8003   (85)     #-8013   (85)     #-9107   (90)
                                                             #-9118   (90)     #-9120   (90)
L_COUNT1                          0000000A-R   162      (2)
L_COUNT3                          0000000E-R   163      (2)  #-5353   (68)     #-5398   (68)     #-5436   (68)     #-5632   (69)
                                                             #-5676   (69)     #-5710   (69)     #-5712   (69)     #-8442   (88)
                                                             #-8487   (88)     #-8525   (88)     #-8721   (89)     #-8765   (89)
                                                             #-8799   (89)     #-8801   (89)
MANUAL                           =00000001     94       (2)
MB_DIAGNOSE_CMD                   00000318-R   1283     (22)  6973    (78)
MB_END_PACKET_1                   00000660-R   2305     (43)  3112    (52)     #-3639   (54)      6884    (77)      6986    (78)
                                                              7088    (79)
MB_END_PACKET_10                  000006DE-R   2457     (46)
```

```
MB_END_PACKET_2                  0000066E-R    2322    (43)     7349    (81)
MB_END_PACKET_3                  0000067C-R    2339    (43)     7423    (81)
MB_END_PACKET_4                  0000068A-R    2356    (44)     2923    (52)     2930    (52)     2941    (52)     2948    (52)
                                                                 3001    (52)     3008    (52)     3015    (52)     3022    (52)
                                                                 3029    (52)     3036    (52)     3043    (52)     3050    (52)
                                                                 3102    (52)     3109    (52)     3119    (52)     7587    (82)
                                                                 7726    (83)     7875    (84)     8015    (85)
MB_END_PACKET_5                  00000698-R    2373    (44)     2954    (52)     2958    (52)     2966    (52)     2970    (52)
                                                                 3066    (52)     3070    (52)     3074    (52)     3078    (52)
                                                                 3082    (52)     3086    (52)     3090    (52)     3094    (52)
                                                                 8160    (86)     8292    (87)     9555    (92)     9693    (93)
MB_END_PACKET_6                  000006A6-R    2390    (44)     2934    (52)     2962    (52)
MB_END_PACKET_7                  000006B4-R    2407    (45)     2983    (52)     2994    (52)
MB_END_PACKET_8                  000006C2-R    2424    (45)
MB_END_PACKET_9                  000006D0-R    2440    (45)     3057    (52)     3062    (52)
MB_INIT_CMD                      000002FC-R    1249    (22)     6871    (77)
MB_NOP_CMD                       0000030A-R    1266    (22)     7075    (79)     9769    (94)
MB_POS_1_CMD                     000005C6-R    2111    (39)     3111    (52)
MB_READ_10_CMD                   0000044C-R    1655    (30)     3003    (52)
MB_READ_11_CMD                   00000468-R    1689    (30)     3017    (52)
MB_READ_12_CMD                   00000484-R    1723    (31)
MB_READ_13_CMD                   000004A0-R    1757    (32)     3031    (52)
MB_READ_14_CMD                   000004BC-R    1791    (32)     3045    (52)
MB_READ_15_CMD                   000004D8-R    1824    (33)
MB_READ_16_CMD                   000004F4-R    1857    (34)     3059    (52)
MB_READ_17_CMD                   00000510-R    1891    (34)     3068    (52)
MB_READ_18_CMD                   0000052C-R    1924    (35)     3076    (52)
MB_READ_19_CMD                   00000548-R    1958    (36)
MB_READ_1_CMD                    00000326-R    1300    (23)     7328    (81)
MB_READ_20_CMD                   00000564-R    1992    (36)     3084    (52)
MB_READ_21_CMD                   00000580-R    2025    (37)     3092    (52)
MB_READ_22_CMD                   0000059C-R    2058    (38)
MB_READ_23_CMD                   000005AA-R    2075    (38)     3097    (52)     3114    (52)
MB_READ_24_CMD                   000005B8-R    2094    (38)     3104    (52)
MB_READ_2_CMD                    00000350-R    1350    (24)     7689    (83)
MB_READ_3_CMD                    0000036C-R    1384    (24)     8257    (87)     9658    (93)
MB_READ_4_CMD                    00000388-R    1417    (25)     7978    (85)
MB_READ_5_CMD                    000003C0-R    1485    (26)     2936    (52)
MB_READ_6_CMD                    000003CE-R    1502    (27)     2943    (52)
MB_READ_7_CMD                    00000406-R    1570    (28)     2964    (52)
MB_READ_8_CMD                    00000414-R    1587    (28)     2968    (52)
MB_READ_9_CMD                    00000430-R    1621    (29)     2985    (52)
MB_WRITE_10_CMD                  000003F8-R    1553    (28)     2960    (52)
MB_WRITE_11_CMD                  00000422-R    1604    (29)     2974    (52)
MB_WRITE_12_CMD                  0000043E-R    1638    (29)     2996    (52)
MB_WRITE_13_CMD                  0000045A-R    1672    (30)     3010    (52)
MB_WRITE_14_CMD                  00000476-R    1706    (31)
MB_WRITE_15_CMD                  00000492-R    1740    (31)     3024    (52)
MB_WRITE_16_CMD                  000004AE-R    1774    (32)     3038    (52)
MB_WRITE_17_CMD                  000004CA-R    1807    (33)
MB_WRITE_18_CMD                  000004E6-R    1841    (33)     3054    (52)
MB_WRITE_19_CMD                  00000502-R    1874    (34)     3064    (52)
MB_WRITE_1_CMD                   00000334-R    1317    (23)     7373    (81)
MB_WRITE_20_CMD                  0000051E-R    1907    (35)     3072    (52)
MB_WRITE_21_CMD                  0000053A-R    1941    (35)
```

```
MB_WRITE_22_CMD                 00000556-R   1975    (36)     3080    (52)
MB_WRITE_23_CMD                 00000572-R   2008    (37)     3088    (52)
MB_WRITE_24_CMD                 0000058E-R   2042    (37)
MB_WRITE_2_CMD                  00000342-R   1334    (23)     7536    (82)
MB_WRITE_3_CMD                  0000035E-R   1367    (24)     8112    (86)       9495    (92)
MB_WRITE_4_CMD                  0000037A-R   1400    (25)     7825    (84)
MB_WRITE_5_CMD                  00000396-R   1434    (25)     2918    (52)
MB_WRITE_6_CMD                  000003A4-R   1451    (26)     2925    (52)
MB_WRITE_7_CMD                  000003B2-R   1468    (26)     2932    (52)
MB_WRITE_8_CMD                  000003DC-R   1519    (27)     2952    (52)
MB_WRITE_9_CMD                  000003EA-R   1536    (27)     2956    (52)
MCHK                           =00000005     94      (2)
MCOMMAND_TBL_1                  000015B3-R   2918    (52)     8443    (88)
MCOMMAND_TBL_2                  0000161F-R   2952    (52)     8722    (89)
MCOMMAND_TBL_3                  0000165B-R   2974    (52)     8989    (90)
MCOMMAND_TBL_4                  0000176B-R   3054    (52)     9264    (91)
MCOMMAND_TBL_5                  000017EB-R   3097    (52)
MINIT_SEQUENCE                  00001FCB-R   3376    (54)     6771    (76)       7162    (80)
MODIFIER_00                    =00000000     123     (2)      1199    (21)       1218    (21)       1235    (22)       1253    (22)
                                                              1270    (22)       1287    (22)       1304    (23)       1321    (23)
                                                              1338    (23)       1354    (24)       1438    (25)       1455    (26)
                                                              1472    (26)       1489    (26)       1506    (27)       1608    (29)
                                                              1625    (29)       1642    (29)       1659    (30)       1676    (30)
                                                              1693    (30)       1710    (31)       1727    (31)       1744    (31)
                                                              1761    (32)       1778    (32)       1795    (32)       1811    (33)
                                                              1828    (33)       2079    (38)       2098    (38)       2115    (39)
                                                              373     (5)        390     (5)        407     (5)        424     (6)
                                                              441     (6)        458     (6)        474     (7)        558     (8)
                                                              575     (9)        592     (9)        609     (9)        626     (10)
                                                              728     (12)       745     (12)       762     (12)       779     (13)
                                                              796     (13)       813     (13)       830     (14)       847     (14)
                                                              864     (14)       881     (15)       898     (15)       915     (15)
                                                              931     (16)       948     (16)
MODIFIER_01                    =00000001     124     (2)      1404    (25)       1421    (25)       524     (8)        541     (8)
MODIFIER_80                    =00000080     125     (2)      1015    (17)       1031    (18)       1048    (18)       1065    (18)
                                                              1082    (19)       1099    (19)       1116    (19)       1132    (20)
                                                              1149    (20)       1166    (20)       1182    (21)       1371    (24)
                                                              1388    (24)       1523    (27)       1540    (27)       1557    (28)
                                                              1574    (28)       1591    (28)       1845    (33)       1861    (34)
                                                              1878    (34)       1895    (34)       1911    (35)       1928    (35)
                                                              1945    (35)       1962    (36)       1979    (36)       1996    (36)
                                                              2012    (37)       2029    (37)       2046    (37)       2062    (38)
                                                              491     (7)        508     (7)        643     (10)       660     (10)
                                                              677     (11)       694     (11)       711     (11)       965     (16)
                                                              981     (17)       998     (17)
MREAD_SERVICE                   0000216C-R   3540    (54)     9649    (93)
MREAD_SERVICE2                  000021AC-R   3571    (54)     9764    (94)
MRECEIVE_TU58                   000020C8-R   3457    (54)     3394    (54)       6883    (77)       6985    (78)       7087    (79)
                                                              7340    (81)       7386    (81)       7422    (81)       7549    (82)
                                                              7586    (82)       7701    (83)       7838    (84)       7874    (84)
                                                              7990    (85)       8124    (86)       8159    (86)       8269    (87)
                                                              8464    (88)       8501    (88)       8547    (88)       8742    (89)
                                                              8778    (89)       8823    (89)       9010    (90)       9051    (90)
                                                              9094    (90)       9284    (91)       9323    (91)       9365    (91)
                                                              9507    (92)       9554    (92)
```

```
MSG_009                         00000002-R   3649   (56)     3668   (56)
MSG_010                         00000002-R   4026   (60)     4059   (60)
MSG_011                         00000002-R   6738   (76)     6757   (76)
MSG_012                         00000002-R   7115   (80)     7148   (80)
MSG_1                           00001AD6-R   3176   (53)     4254   (61)     7343   (81)
MSG_2                           00001AE6-R   3181   (53)
M_CLOCK_FLAG                   =00000001     172    (2)    #-3522   (54)
M_DATA_FLAG                    =00000004     176    (2)    #-4617   (63)   #-4906   (65)   #-5185   (67)   #-5463   (68)
                                                           #-5739   (69)   #-6010   (70)   #-6281   (71)   #-6590   (73)
                                                           #-7706   (83)   #-7995   (85)   #-8274   (87)   #-8552   (88)
                                                           #-8828   (89)   #-9099   (90)   #-9370   (91)   #-9678   (93)
M_ENABLE                       =00000040     141    (2)    #-3420   (54)   #-3443   (54)   #-3462   (54)   #-6447   (72)
                                                           #-6448   (72)   #-6567   (73)   #-6582   (73)   #-6696   (74)
                                                           #-6697   (74)   #-6712   (74)   #-6713   (74)   #-9536   (92)
                                                           #-9537   (92)   #-9655   (93)   #-9670   (93)   #-9784   (94)
                                                           #-9785   (94)   #-9801   (94)   #-9802   (94)
M_END_FLAG                     =00000008     178    (2)    #-3799   (57)   #-3901   (58)   #-4003   (59)   #-4339   (61)
                                                           #-4502   (62)   #-4790   (64)   #-5075   (66)   #-5417   (68)
                                                           #-5694   (69)   #-5967   (70)   #-6239   (71)   #-6470   (72)
                                                           #-6888   (77)   #-6990   (78)   #-7092   (79)   #-7428   (81)
                                                           #-7591   (82)   #-7879   (84)   #-8164   (86)   #-8506   (88)
                                                           #-8783   (89)   #-9056   (90)   #-9328   (91)   #-9559   (92)
M_ERROR                        =80000000     144    (2)    #-3420   (54)   #-3424   (54)   #-3443   (54)   #-3447   (54)
                                                           #-3462   (54)   #-3466   (54)   #-3521   (54)   #-6447   (72)
                                                           #-6582   (73)   #-6696   (74)   #-6712   (74)   #-9536   (92)
                                                           #-9670   (93)   #-9784   (94)   #-9801   (94)
M_INTERRUPT                    =00000080     145    (2)    #-3424   (54)   #-3447   (54)   #-3466   (54)   #-3521   (54)
M_RD_INT_FLAG                  =00000004     188    (2)    #-3562   (54)   #-3572   (54)
M_RECEIVE_FLAG                 =00000002     174    (2)    #-3363   (54)   #-3397   (54)
M_REPEAT_CHECK                 =00000004     184    (2)    #-5898   (70)   #-6173   (71)   #-8987   (90)   #-9262   (91)
M_REPEAT_SEND                  =00000002     182    (2)    #-3705   (56)   #-4096   (60)   #-5898   (70)   #-5956   (70)
                                                           #-6173   (71)   #-6228   (71)   #-6794   (76)   #-7185   (80)
                                                           #-8987   (90)   #-9045   (90)   #-9262   (91)   #-9317   (91)
M_RUN                          =00000001     143    (2)    #-3420   (54)   #-3443   (54)   #-3462   (54)   #-6447   (72)
                                                           #-6582   (73)   #-6696   (74)   #-6712   (74)   #-9536   (92)
                                                           #-9670   (93)   #-9784   (94)   #-9801   (94)
M_SDONE_FLAG                   =00000002     186    (2)    #-3596   (54)
M_SEND_FLAG                    =00000010     180    (2)    #-3366   (54)   #-3400   (54)
M_TRANSFER                     =00000010     140    (2)    #-3416   (54)   #-3441   (54)   #-3460   (54)   #-6445   (72)
                                                           #-6580   (73)   #-6694   (74)   #-6710   (74)   #-9534   (92)
                                                           #-9668   (93)   #-9782   (94)   #-9799   (94)
NOT_USED                       =00000000     130    (2)      2141   (40)     2158   (40)     2175   (40)     2192   (41)
                                                             2209   (41)     2226   (41)     2243   (42)     2260   (42)
                                                             2276   (42)     2293   (43)
NO_INT_ERR                      00001F4B-R   3319   (53)     6719   (74)     9808   (94)
NO_OPERATION                   =00000000     112    (2)      1269   (22)      389   (5)
NO_READY                        00001BEB-R   3198   (53)     3687   (56)     3786   (57)     3888   (58)     3990   (59)
                                                             4078   (60)     4243   (61)     4288   (61)     4324   (61)
                                                             4451   (62)     4486   (62)     4604   (63)     4740   (64)
                                                             4775   (64)     4893   (65)     5027   (66)     5061   (66)
                                                             5172   (67)     5366   (68)     5401   (68)     5450   (68)
                                                             5645   (69)     5679   (69)     5726   (69)     5912   (70)
                                                             5947   (70)     5994   (70)     6187   (71)     6221   (71)
                                                             6265   (71)     6410   (72)     6574   (73)     6685   (74)
                                                             6776   (76)     6875   (77)     6977   (78)     7079   (79)
```

```
                                                                  7167    (80)       7332    (81)       7377    (81)       7413    (81)
                                                                  7540    (82)       7575    (82)       7693    (83)       7829    (84)
                                                                  7864    (84)       7982    (85)       8116    (86)       8150    (86)
                                                                  8261    (87)       8455    (88)       8490    (88)       8539    (88)
                                                                  8734    (89)       8768    (89)       8815    (89)       9001    (90)
                                                                  9036    (90)       9083    (90)       9276    (91)       9310    (91)
                                                                  9354    (91)       9499    (92)       9662    (93)       9773    (94)
PAR$M_ATDEF                     =00000010      92      (2)
PAR$M_ATHI                      =00000008      92      (2)
PAR$M_ATLO                      =00000004      92      (2)
PAR$M_NODEF                     =00000002      92      (2)
PAR$V_ATDEF                     =00000004      92      (2)
PAR$V_ATHI                      =00000003      92      (2)
PAR$V_ATLO                      =00000002      92      (2)
PAR$V_NODEF                     =00000001      92      (2)
PAR$_BIN                        =00000002      92      (2)
PAR$_DEC                        =0000000A      92      (2)
PAR$_HEX                        =00000010      92      (2)
PAR$_NO                         =00000000      92      (2)      #-4280    (61)     #-7369    (81)
PAR$_OCT                        =00000008      92      (2)
PAR$_YES                        =00000001      92      (2)
PCS                             =00000008      94      (2)
POSITION                        =00000005      116     (2)        1234    (22)       2114    (39)
POWER_FAIL                      =00000003      94      (2)
PR$_ICCS                         00000000-XR                    #-3416    (54)     #-3420    (54)     #-3424    (54)     #-3441    (54)
                                                                #-3443    (54)     #-3447    (54)     #-3460    (54)     #-3462    (54)
                                                                #-3466    (54)     #-3520    (54)     #-3521    (54)     #-6445    (72)
                                                                #-6447    (72)     #-6580    (73)     #-6582    (73)     #-6694    (74)
                                                                #-6696    (74)     #-6710    (74)     #-6712    (74)     #-9534    (92)
                                                                #-9536    (92)     #-9668    (93)     #-9670    (93)     #-9782    (94)
                                                                #-9784    (94)     #-9799    (94)     #-9801    (94)
PR750$_CSRD                     =0000001D                       #-3350    (54)     #-3364    (54)     #-3384    (54)     #-3398    (54)
                                                                #-3448    (54)     #-3467    (54)     #-3532    (54)     #-3543    (54)
                                                                #-3575    (54)     #-6566    (73)     #-6691    (74)     #-9779    (94)
PR750$_CSRS                     =0000001C                       #-3444    (54)     #-3463    (54)     #-3541    (54)     #-3563    (54)
                                                                #-3573    (54)     #-6567    (73)     #-6585    (73)     #-6697    (74)
                                                                #-6701    (74)     #-6707    (74)     #-6713    (74)     #-6718    (74)
                                                                #-6724    (74)     #-9655    (93)     #-9673    (93)     #-9785    (94)
                                                                #-9789    (94)     #-9796    (94)     #-9802    (94)     #-9807    (94)
                                                                #-9813    (94)
PR750$_CSTD                     =0000001F                       #-3419    (54)     #-3549    (54)     #-3579    (54)     #-3592    (54)
PR750$_CSTS                     =0000001E                       #-3347    (54)     #-3381    (54)     #-3422    (54)     #-3550    (54)
                                                                #-3580    (54)     #-3597    (54)     #-6448    (72)     #-6453    (72)
                                                                #-9537    (92)     #-9542    (92)
PR750$_NICR                     =00000019                       #-3415    (54)     #-3417    (54)     #-3440    (54)     #-3442    (54)
                                                                #-3459    (54)     #-3461    (54)     #-6444    (72)     #-6446    (72)
                                                                #-6579    (73)     #-6581    (73)     #-6693    (74)     #-6695    (74)
                                                                #-6709    (74)     #-6711    (74)     #-9533    (92)     #-9535    (92)
                                                                #-9667    (93)     #-9669    (93)     #-9781    (94)     #-9783    (94)
                                                                #-9798    (94)     #-9800    (94)
READ                            =00000002      114     (2)        1014    (17)       1047    (18)       1081    (19)       1115    (19)
                                                                  1148    (20)       1181    (21)       1198    (21)       1217    (21)
                                                                  1303    (23)       1353    (24)       1387    (24)       1420    (25)
                                                                  1488    (26)       1505    (27)       1573    (28)       1590    (28)
                                                                  1624    (29)       1658    (30)       1692    (30)       1726    (31)
```

```
                                                 1760  (32)      1794  (32)      1827  (33)      1860  (34)
                                                 1894  (34)      1927  (35)      1961  (36)      1995  (36)
                                                 2028  (37)      2061  (38)      2078  (38)      2097  (38)
                                                  423  (6)        473  (7)        507  (7)        540  (8)
                                                  608  (9)        625  (10)       693  (11)       710  (11)
                                                  744  (12)       778  (13)       812  (13)       846  (14)
                                                  880  (15)       914  (15)       947  (16)       980  (17)
READ_SERVICE        00002164-R   3531   (54)     6560  (73)
READ_SERVICE2       000021A0-R   3561   (54)     6676  (74)
RECEIVE_TU58        00002083-R   3438   (54)     3360  (54)      3794  (57)      3896  (58)      3998  (59)
                                                 4251  (61)      4297  (61)      4333  (61)      4460  (62)
                                                 4497  (62)      4612  (63)      4749  (64)      4785  (64)
                                                 4901  (65)      5035  (66)      5070  (66)      5180  (67)
                                                 5375  (68)      5412  (68)      5458  (68)      5653  (69)
                                                 5689  (69)      5734  (69)      5921  (70)      5962  (70)
                                                 6005  (70)      6195  (71)      6234  (71)      6276  (71)
                                                 6418  (72)      6465  (72)
REC_ERR             00001EB4-R   3288   (53)     3695  (56)      3801  (57)      3903  (58)      4005  (59)
                                                 4086  (60)      4304  (61)      4341  (61)      4466  (62)
                                                 4504  (62)      4619  (63)      4755  (64)      4792  (64)
                                                 4908  (65)      5041  (66)      5077  (66)      5187  (67)
                                                 5381  (68)      5419  (68)      5465  (68)      5659  (69)
                                                 5696  (69)      5741  (69)      5927  (70)      5969  (70)
                                                 6012  (70)      6201  (71)      6241  (71)      6283  (71)
                                                 6424  (72)      6472  (72)      6593  (73)      6784  (76)
                                                 6890  (77)      6992  (78)      7094  (79)      7175  (80)
                                                 7393  (81)      7430  (81)      7555  (82)      7593  (82)
                                                 7708  (83)      7844  (84)      7881  (84)      7997  (85)
                                                 8130  (86)      8166  (86)      8276  (87)      8470  (88)
                                                 8508  (88)      8554  (88)      8748  (89)      8785  (89)
                                                 8830  (89)      9016  (90)      9058  (90)      9101  (90)
                                                 9290  (91)      9330  (91)      9372  (91)      9513  (92)
                                                 9561  (92)      9681  (93)
SCB$L_CSRD          =000000F0    153    (2)    #-6560  (73)    #-6675  (74)    #-6676  (74)    #-6730  (74)
                                               #-9649  (93)    #-9763  (94)    #-9764  (94)    #-9819  (94)
SCB$L_CSTD          =000000F4    152    (2)    #-6397  (72)    #-6731  (74)    #-9486  (92)    #-9820  (94)
SCB$L_TIMER          000000C0                  #-3674  (56)    #-4020  (59)    #-4064  (60)    #-6729  (74)
                                               #-6763  (76)    #-7109  (79)    #-7153  (80)    #-9818  (94)
SEND_INT_ERR        00001AC9-R   3171   (53)     6456  (72)      9545  (92)
SEND_TU58           0000203D-R   3413   (54)     3348  (54)      3355  (54)      3382  (54)      3389  (54)
                                                 3473  (54)      3784  (57)      3886  (58)      3988  (59)
                                                 4241  (61)      4286  (61)      4322  (61)      4449  (62)
                                                 4484  (62)      4602  (63)      4738  (64)      4773  (64)
                                                 4891  (65)      5025  (66)      5059  (66)      5170  (67)
                                                 5364  (68)      5399  (68)      5448  (68)      5643  (69)
                                                 5677  (69)      5724  (69)      5910  (70)      5945  (70)
                                                 5992  (70)      6185  (71)      6219  (71)      6263  (71)
                                                 6408  (72)      6572  (73)      6683  (74)      6873  (77)
                                                 6975  (78)      7077  (79)      7330  (81)      7375  (81)
                                                 7411  (81)      7538  (82)      7573  (82)      7691  (83)
                                                 7827  (84)      7862  (84)      7980  (85)      8114  (86)
                                                 8148  (86)      8259  (87)      8453  (88)      8488  (88)
                                                 8537  (88)      8732  (89)      8766  (89)      8813  (89)
                                                 8999  (90)      9034  (90)      9081  (90)      9274  (91)
                                                 9308  (91)      9352  (91)      9497  (92)      9660  (93)
```

```
                                                      9771    (94)
SEP_FUNCTIONAL                 =00000002     72      (2)
SEP_REPAIR                     =00000003     72      (2)
SEQUENCE_H                     =00000000     128     (2)     1002    (17)        1019    (17)        1035    (18)        1052    (18)
                                                      1069    (18)        1086    (19)        1103    (19)        1120    (19)
                                                      1136    (20)        1153    (20)        1170    (20)        1186    (21)
                                                      1203    (21)        1222    (21)        1239    (22)        1257    (22)
                                                      1274    (22)        1291    (22)        1308    (23)        1325    (23)
                                                      1342    (23)        1358    (24)        1375    (24)        1392    (24)
                                                      1408    (25)        1425    (25)        1442    (25)        1459    (26)
                                                      1476    (26)        1493    (26)        1510    (27)        1527    (27)
                                                      1544    (27)        1561    (28)        1578    (28)        1595    (28)
                                                      1612    (29)        1629    (29)        1646    (29)        1663    (30)
                                                      1680    (30)        1697    (30)        1714    (31)        1731    (31)
                                                      1748    (31)        1765    (32)        1782    (32)        1799    (32)
                                                      1815    (33)        1832    (33)        1849    (33)        1865    (34)
                                                      1882    (34)        1899    (34)        1915    (35)        1932    (35)
                                                      1949    (35)        1966    (36)        1983    (36)        2000    (36)
                                                      2016    (37)        2033    (37)        2050    (37)        2066    (38)
                                                      2083    (38)        2102    (38)        2119    (39)        2143    (40)
                                                      2160    (40)        2177    (40)        2194    (41)        2211    (41)
                                                      2228    (41)        2245    (42)        2262    (42)        2278    (42)
                                                      2295    (43)        2313    (43)        2330    (43)        2347    (43)
                                                      2364    (44)        2381    (44)        2398    (44)        2415    (45)
                                                      2432    (45)        2448    (45)        2465    (46)        377     (5)
                                                      394     (5)         411     (5)         428     (6)         445     (6)
                                                      462     (6)         478     (7)         495     (7)         512     (7)
                                                      528     (8)         545     (8)         562     (8)         579     (9)
                                                      596     (9)         613     (9)         630     (10)        647     (10)
                                                      664     (10)        681     (11)        698     (11)        715     (11)
                                                      732     (12)        749     (12)        766     (12)        783     (13)
                                                      800     (13)        817     (13)        834     (14)        851     (14)
                                                      868     (14)        885     (15)        902     (15)        919     (15)
                                                      935     (16)        952     (16)        969     (16)        985     (17)
SEQUENCE_L                     =00000000     129     (2)     1001    (17)        1018    (17)        1034    (18)        1051    (18)
                                                      1068    (18)        1085    (19)        1102    (19)        1119    (19)
                                                      1135    (20)        1152    (20)        1169    (20)        1185    (21)
                                                      1202    (21)        1221    (21)        1238    (22)        1256    (22)
                                                      1273    (22)        1290    (22)        1307    (23)        1324    (23)
                                                      1341    (23)        1357    (24)        1374    (24)        1391    (24)
                                                      1407    (25)        1424    (25)        1441    (25)        1458    (26)
                                                      1475    (26)        1492    (26)        1509    (27)        1526    (27)
                                                      1543    (27)        1560    (28)        1577    (28)        1594    (28)
                                                      1611    (29)        1628    (29)        1645    (29)        1662    (30)
                                                      1679    (30)        1696    (30)        1713    (31)        1730    (31)
                                                      1747    (31)        1764    (32)        1781    (32)        1798    (32)
                                                      1814    (33)        1831    (33)        1848    (33)        1864    (34)
                                                      1881    (34)        1898    (34)        1914    (35)        1931    (35)
                                                      1948    (35)        1965    (36)        1982    (36)        1999    (36)
                                                      2015    (37)        2032    (37)        2049    (37)        2065    (38)
                                                      2082    (38)        2101    (38)        2118    (39)        2142    (40)
                                                      2159    (40)        2176    (40)        2193    (41)        2210    (41)
                                                      2227    (41)        2244    (42)        2261    (42)        2277    (42)
                                                      2294    (43)        2312    (43)        2329    (43)        2346    (43)
                                                      2363    (44)        2380    (44)        2397    (44)        2414    (45)
```

```
                                                       2431    (45)    2447    (45)    2464    (46)    376     (5)
                                                       393     (5)     410     (5)     427     (6)     444     (6)
                                                       461     (6)     477     (7)     494     (7)     511     (7)
                                                       527     (8)     544     (8)     561     (8)     578     (9)
                                                       595     (9)     612     (9)     629     (10)    646     (10)
                                                       663     (10)    680     (11)    697     (11)    714     (11)
                                                       731     (12)    748     (12)    765     (12)    782     (13)
                                                       799     (13)    816     (13)    833     (14)    850     (14)
                                                       867     (14)    884     (15)    901     (15)    918     (15)
                                                       934     (16)    951     (16)    968     (16)    984     (17)
SIZ...                        =00000001   92     (2)   72      (2)     92      (2)
SUCCESS_CODE                  =00000000   122    (2)   2139    (40)    2156    (40)    2173    (40)    2190    (41)
                                                       2207    (41)    2224    (41)    2241    (42)    2258    (42)
                                                       2274    (42)    2291    (43)    2309    (43)    2326    (43)
                                                       2343    (43)    2360    (44)    2377    (44)    2394    (44)
                                                       2411    (45)    2428    (45)    2444    (45)    2461    (46)
SUMMARY_STAT_H                =00000000   132    (2)   2147    (40)    2164    (40)    2181    (40)    2198    (41)
                                                       2215    (41)    2232    (41)    2249    (42)    2266    (42)
                                                       2282    (42)    2299    (43)    2317    (43)    2334    (43)
                                                       2351    (43)    2368    (44)    2385    (44)    2402    (44)
                                                       2419    (45)    2436    (45)    2452    (45)    2469    (46)
SUMMARY_STAT_L                =00000000   133    (2)   2146    (40)    2163    (40)    2180    (40)    2197    (41)
                                                       2214    (41)    2231    (41)    2248    (42)    2265    (42)
                                                       2281    (42)    2298    (43)    2316    (43)    2333    (43)
                                                       2350    (43)    2367    (44)    2384    (44)    2401    (44)
                                                       2418    (45)    2435    (45)    2451    (45)    2468    (46)
SWITCH_00                     =00000000   126    (2)   1000    (17)    1017    (17)    1033    (18)    1050    (18)
                                                       1067    (18)    1084    (19)    1101    (19)    1118    (19)
                                                       1134    (20)    1151    (20)    1168    (20)    1184    (21)
                                                       1201    (21)    1220    (21)    1237    (22)    375     (5)
                                                       392     (5)     409     (5)     426     (6)     443     (6)
                                                       460     (6)     476     (7)     493     (7)     510     (7)
                                                       526     (8)     543     (8)     560     (8)     577     (9)
                                                       594     (9)     611     (9)     628     (10)    645     (10)
                                                       662     (10)    679     (11)    696     (11)    713     (11)
                                                       730     (12)    747     (12)    764     (12)    781     (13)
                                                       798     (13)    815     (13)    832     (14)    849     (14)
                                                       866     (14)    883     (15)    900     (15)    917     (15)
                                                       933     (16)    950     (16)    967     (16)    983     (17)
SWITCH 08                     -00000008   127    (2)   1255    (22)    1272    (22)    1289    (22)    1306    (23)
                                                       1323    (23)    1340    (23)    1356    (24)    1373    (24)
                                                       1390    (24)    1406    (25)    1423    (25)    1440    (25)
                                                       1457    (26)    1474    (26)    1491    (26)    1508    (27)
                                                       1525    (27)    1542    (27)    1559    (28)    1576    (28)
                                                       1593    (28)    1610    (29)    1627    (29)    1644    (29)
                                                       1661    (30)    1678    (30)    1695    (30)    1712    (31)
                                                       1729    (31)    1746    (31)    1763    (32)    1780    (32)
                                                       1797    (32)    1813    (33)    1830    (33)    1847    (33)
                                                       1863    (34)    1880    (34)    1897    (34)    1913    (35)
                                                       1930    (35)    1947    (35)    1964    (36)    1981    (36)
                                                       1998    (36)    2014    (37)    2031    (37)    2048    (37)
                                                       2064    (38)    2081    (38)    2100    (38)    2117    (39)
                                                       2311    (43)    2328    (43)    2345    (43)    2362    (44)
                                                       2379    (44)    2396    (44)    2413    (45)    2430    (45)
                                                       2446    (45)    2463    (46)
```

```
T10_S1                          00000131-R    4231     (61)
T10_S10                         000014D8-R    5896     (70)
T10_S10_X                       00001854-R    6046     (70)
T10_S11                         0000185F-R    6171     (71)
T10_S11_X                       00001BB1-R    6315     (71)
T10_S12                         00001BBC-R    6396     (72)
T10_S12_X                       00001DEB-R    6487     (72)
T10_S13                         00001DF6-R    6559     (73)
T10_S13_X                       00001F86-R    6616     (73)
T10_S14                         00001F91-R    6674     (74)
T10_S14_X                       00002105-R    6727     (74)
T10_S1_X                        00000436-R    4358     (61)    #-4279    (61)
T10_S2                          00000441-R    4439     (62)
T10_S2_X                        00000625-R    4519     (62)
T10_S3                          00000630-R    4592     (63)
T10_S3_X                        00000786-R    4648     (63)
T10_S4                          00000791-R    4728     (64)
T10_S4_X                        00000975-R    4807     (64)
T10_S5                          00000980-R    4881     (65)
T10_S5_X                        00000AD6-R    4935     (65)
T10_S6                          00000AE1-R    5014     (66)
T10_S6_X                        00000CBB-R    5092     (66)
T10_S7                          00000CC6-R    5160     (67)
T10_S7_X                        00000E0E-R    5213     (67)
T10_S8                          00000E19-R    5350     (68)
T10_S8_X                        00001186-R    5497     (68)
T10_S9                          00001191-R    5629     (69)
T10_S9_X                        000014CD-R    5771     (69)
T11_S1                          00000135-R    6864     (77)
T11_S1_X                        0000022D-R    6904     (77)
T11_S2                          00000238-R    6965     (78)
T11_S2_X                        00000330-R    7006     (78)
T11_S3                          0000033B-R    7067     (79)
T11_S3_X                        00000433-R    7108     (79)
T12_S1                          0000013D-R    7320     (81)
T12_S10                         000014E4-R    8985     (90)
T12_S10_X                       00001860-R    9135     (90)
T12_S11                         0000186B-R    9260     (91)
T12_S11_X                       00001BBD-R    9404     (91)
T12_S12                         00001BC8-R    9485     (92)
T12_S12_X                       00001DF7-R    9576     (92)
T12_S13                         00001E02-R    9648     (93)
T12_S13_X                       00001F8F-R    9704     (93)
T12_S14                         00001F9A-R    9762     (94)
T12_S14_X                       00002115-R    9816     (94)
T12_S1_X                        00000442-R    7447     (81)    #-7368    (81)
T12_S2                          0000044D-R    7528     (82)
T12_S2_X                        00000631-R    7608     (82)
T12_S3                          0000063C-R    7681     (83)
T12_S3_X                        00000792-R    7737     (83)
T12_S4                          0000079D-R    7817     (84)
T12_S4_X                        00000981-R    7896     (84)
T12_S5                          0000098C-R    7970     (85)
T12_S5_X                        00000AE2-R    8024     (85)
T12_S6                          00000AED-R    8103     (86)
```

```
T12_S6_X                       00000CC7-R   8181  (86)
T12_S7                         00000CD2-R   8249  (87)
T12_S7_X                       00000E1A-R   8302  (87)
T12_S8                         00000E25-R   8439  (88)
T12_S8_X                       00001192-R   8586  (88)
T12_S9                         0000119D-R   8718  (89)
T12_S9_X                       000014D9-R   8860  (89)
T9_S1                          00000129-R   3775  (57)
T9_S1_X                        00000221-R   3815  (57)
T9_S2                          0000022C-R   3876  (58)
T9_S2_X                        00000324-R   3917  (58)
T9_S3                          0000032F-R   3978  (59)
T9_S3_X                        00000427-R   4019  (59)
TEMP5                          00000017-R   166   (2)    #-3468   (54)  #-3475   (54)  #-3545   (54)  #-3552   (54)
TEMP7                          0000001B-R   167   (2)    #-3469   (54)  #-3474   (54)  #-3546   (54)  #-3553   (54)
TEST_009                       00000018-R   3668  (56)     3668   (56)
TEST_009_X                     00000442-R   4022  (59)
TEST_010                       00000020-R   4059  (60)     4059   (60)
TEST_010_X                     0000213A-R   6733  (74)   #-4281   (61)
TEST_011                       00000024-R   6757  (76)     6757   (76)
TEST_011_X                     0000044E-R   7111  (79)
TEST_012                       0000002C-R   7148  (80)     7148   (80)
TEST_012_X                     0000214A-R   9822  (94)   #-7370   (81)
TIMER_SERVICE                  00002150-R   3519  (54)     3674   (56)    4064   (60)    6763   (76)    7153   (80)
TU58                          =00000002     94    (2)      3668   (56)    4059   (60)    6757   (76)    7148   (80)
T_ACT_CONT                     00001C68-R   3215  (53)     3264   (53)
T_ACT_DATA                     00001C9B-R   3221  (53)     3271   (53)
T_ACT_END                      00001DB5-R   3248  (53)     3306   (53)
T_ACT_REC                      00001E18-R   3254  (53)     3310   (53)
T_BAD_CONT                     00001BF8-R   3203  (53)     3262   (53)
T_BAD_DATA                     00001C37-R   3209  (53)     3269   (53)
T_BAD_END                      00001C19-R   3206  (53)     3276   (53)
T_DUMP_14                      00001E42-R   3257  (53)     3626   (54)
T_DUMP_HDR                     00001CBE-R   3224  (53)     3610   (54)
T_DUMP_NOXOR                   00001CF2-R   3230  (53)     3616   (54)
T_DUMP_XOR                     00001CDA-R   3227  (53)     3614   (54)
T_EMPTY_BUF                    00001D93-R   3245  (53)     3300   (53)
T_EXP_CONT                     00001C56-R   3212  (53)     3263   (53)
T_EXP_DATA                     00001C7A-R   3218  (53)     3270   (53)
T_INIT1_ERR                    0000183B-R   3124  (53)     3687   (56)    3695   (56)    3708   (56)    4078   (60)
                                                           4086   (60)    4099   (60)    6776   (76)    6784   (76)
                                                           6797   (76)    7167   (80)    7175   (80)    7188   (80)
T_INIT2_ERR                    00001865-R   3127  (53)     3786   (57)    3801   (57)    3810   (57)    6875   (77)
                                                           6890   (77)    6899   (77)
T_INT_ERR                      00001A2C-R   3163  (53)     3316   (53)
T_IPL_ERR                      00001A04-R   3160  (53)     6685   (74)    6702   (74)    6719   (74)    9773   (94)
                                                           9790   (94)    9808   (94)
T_MSG_1                        00001AF6-R   3186  (53)     3178   (53)
T_MSG_2                        00001B28-R   3189  (53)     3183   (53)
T_NOP_ERR                      000018AD-R   3133  (53)     3990   (59)    4005   (59)    4014   (59)    7079   (79)
                                                           7094   (79)    7103   (79)
T_NO_INT_ERR                   00001A65-R   3166  (53)     3321   (53)
T_NO_READY                     00001BB9-R   3196  (53)     3200   (53)
T_OVERWRITE                    00001B56-R   3192  (53)     4280   (61)    7369   (81)
T_RD_BLK_ERR                   000018EB-R   3139  (53)     4604   (63)    4619   (63)    4630   (63)    4642   (63)
```

```
                                                   5450    (68)      5465    (68)      5476    (68)      5488    (68)
                                                   5994    (70)      6012    (70)      6023    (70)      6039    (70)
                                                   7693    (83)      7708    (83)      7719    (83)      7731    (83)
                                                   8539    (88)      8554    (88)      8565    (88)      8577    (88)
                                                   9083    (90)      9101    (90)      9112    (90)      9128    (90)
T_RD_INT_ERR                   0000199F-R   3154    (53)   6574    (73)      6593    (73)      6601    (73)      6611    (73)
                                                   9662    (93)      9681    (93)      9689    (93)      9699    (93)
T_RD_MOD_ERR                   0000197A-R   3151    (53)   4893    (65)      4908    (65)      4919    (65)      4931    (65)
                                                   7982    (85)      7997    (85)      8008    (85)      8020    (85)
T_RD_REC_ERR                   00001931-R   3145    (53)   4243    (61)      4254    (61)      4267    (61)      5172    (67)
                                                   5187    (67)      5197    (67)      5208    (67)      5726    (69)
                                                   5741    (69)      5751    (69)      5762    (69)      6265    (71)
                                                   6283    (71)      6293    (71)      6308    (71)      7332    (81)
                                                   7343    (81)      7356    (81)      8261    (87)      8276    (87)
                                                   8286    (87)      8297    (87)      8815    (89)      8830    (89)
                                                   8840    (89)      8851    (89)      9354    (91)      9372    (91)
                                                   9382    (91)      9397    (91)
T_REC_CONT                     00001D47-R   3239    (53)  #-3695    (56)     #-4086    (60)     #-4304    (61)     #-4466    (62)
                                                  #-4755    (64)     #-5041    (66)     #-5381    (68)     #-5659    (69)
                                                  #-5927    (70)     #-6201    (71)     #-6424    (72)     #-6784    (76)
                                                  #-7175    (80)     #-7393    (81)     #-7555    (82)     #-7844    (84)
                                                  #-8130    (86)     #-8470    (88)     #-8748    (89)     #-9016    (90)
                                                  #-9290    (91)     #-9513    (92)
T_REC_DATA                     00001D68-R   3242    (53)  #-4619    (63)     #-4908    (65)     #-5187    (67)     #-5465    (68)
                                                  #-5741    (69)     #-6012    (70)     #-6283    (71)     #-6593    (73)
                                                  #-7708    (83)     #-7997    (85)     #-8276    (87)     #-8554    (88)
                                                  #-8830    (89)     #-9101    (90)     #-9372    (91)     #-9681    (93)
T_REC_END                      00001D2E-R   3236    (53)  #-3801    (57)     #-3903    (58)     #-4005    (59)     #-4341    (61)
                                                  #-4504    (62)     #-4792    (64)     #-5077    (66)     #-5419    (68)
                                                  #-5696    (69)     #-5969    (70)     #-6241    (71)     #-6472    (72)
                                                  #-6890    (77)     #-6992    (78)     #-7094    (79)     #-7430    (81)
                                                  #-7593    (82)     #-7881    (84)     #-8166    (86)     #-8508    (88)
                                                  #-8785    (89)     #-9058    (90)     #-9330    (91)     #-9561    (92)
T_REC_ERR                      00001D04-R   3233    (53)   3290    (53)
T_SEND_INT_ERR                 00001A89-R   3169    (53)   3173    (53)
T_SLF_DIAG_ERR                 00001888-R   3130    (53)   3888    (58)      3903    (58)      3912    (58)      6977    (78)
                                                   6992    (78)      7001    (78)
T_UNREC_DATA                   00001DF4-R   3251    (53)   3309    (53)
T_WR_BLK_ERR                   000018C8-R   3136    (53)   4451    (62)      4466    (62)      4476    (62)      4486    (62)
                                                   4504    (62)      4514    (62)      5366    (68)      5381    (68)
                                                   5391    (68)      5401    (68)      5419    (68)      5429    (68)
                                                   5912    (70)      5927    (70)      5937    (70)      5947    (70)
                                                   5969    (70)      5979    (70)      7540    (82)      7555    (82)
                                                   7565    (82)      7575    (82)      7593    (82)      7603    (82)
                                                   8455    (88)      8470    (88)      8480    (88)      8490    (88)
                                                   8508    (88)      8518    (88)      9001    (90)      9016    (90)
                                                   9026    (90)      9036    (90)      9058    (90)      9068    (90)
T_WR_INT_ERR                   000019D1-R   3157    (53)   6410    (72)      6424    (72)      6434    (72)      6456    (72)
                                                   6472    (72)      6482    (72)      9499    (92)      9513    (92)
                                                   9523    (92)      9545    (92)      9561    (92)      9571    (92)
T_WR_MOD_ERR                   00001954-R   3148    (53)   4740    (64)      4755    (64)      4765    (64)      4775    (64)
                                                   4792    (64)      4802    (64)      7829    (84)      7844    (84)
                                                   7854    (84)      7864    (84)      7881    (84)      7891    (84)
T_WR_REC ERR                   0000190D-R   3142    (53)   4288    (61)      4304    (61)      4314    (61)      4324    (61)
                                                   4341    (61)      4351    (61)      5027    (66)      5041    (66)
```

```
                                                              5051   (66)     5061   (66)     5077   (66)     5087   (66)
                                                              5645   (69)     5659   (69)     5669   (69)     5679   (69)
                                                              5696   (69)     5706   (69)     6187   (71)     6201   (71)
                                                              6211   (71)     6221   (71)     6241   (71)     6251   (71)
                                                              7377   (81)     7393   (81)     7403   (81)     7413   (81)
                                                              7430   (81)     7440   (81)     8116   (86)     8130   (86)
                                                              8140   (86)     8150   (86)     8166   (86)     8176   (86)
                                                              8734   (89)     8748   (89)     8758   (89)     8768   (89)
                                                              8785   (89)     8795   (89)     9276   (91)     9290   (91)
                                                              9300   (91)     9310   (91)     9330   (91)     9340   (91)
UNIT 0                          =00000000     131    (2)      1016   (17)     1032   (18)     1049   (18)     1066   (18)
                                                              1083   (19)     1100   (19)     1117   (19)     1133   (20)
                                                              1150   (20)     1167   (20)     1183   (21)     1200   (21)
                                                              1219   (21)     1236   (22)     1254   (22)     1271   (22)
                                                              1288   (22)     1305   (23)     1322   (23)     1339   (23)
                                                              1355   (24)     1372   (24)     1389   (24)     1405   (25)
                                                              1422   (25)     1439   (25)     1456   (26)     1473   (26)
                                                              1490   (26)     1507   (27)     1524   (27)     1541   (27)
                                                              1558   (28)     1575   (28)     1592   (28)     1609   (29)
                                                              1626   (29)     1643   (29)     1660   (30)     1677   (30)
                                                              1694   (30)     1711   (31)     1728   (31)     1745   (31)
                                                              1762   (32)     1779   (32)     1796   (32)     1812   (33)
                                                              1829   (33)     1846   (33)     1862   (34)     1879   (34)
                                                              1896   (34)     1912   (35)     1929   (35)     1946   (35)
                                                              1963   (36)     1980   (36)     1997   (36)     2013   (37)
                                                              2030   (37)     2047   (37)     2063   (38)     2080   (38)
                                                              2099   (38)     2116   (39)     2140   (40)     2157   (40)
                                                              2174   (40)     2191   (41)     2208   (41)     2225   (41)
                                                              2242   (42)     2259   (42)     2275   (42)     2292   (43)
                                                              2310   (43)     2327   (43)     2344   (43)     2361   (44)
                                                              2378   (44)     2395   (44)     2412   (45)     2429   (45)
                                                              2445   (45)     2462   (46)     374    (5)      391    (5)
                                                              408    (5)      425    (6)      442    (6)      459    (6)
                                                              475    (7)      492    (7)      509    (7)      525    (8)
                                                              542    (8)      559    (8)      576    (9)      593    (9)
                                                              610    (9)      627    (10)     644    (10)     661    (10)
                                                              678    (11)     695    (11)     712    (11)     729    (12)
                                                              746    (12)     763    (12)     780    (13)     797    (13)
                                                              814    (13)     831    (14)     848    (14)     865    (14)
                                                              882    (15)     899    (15)     916    (15)     932    (16)
                                                              949    (16)     966    (16)     982    (17)     999    (17)
V CLOCK FLAG                    =00000000     171    (2)    #-3349   (54)   #-3356   (54)   #-3383   (54)   #-3390   (54)
                                                            #-3421   (54)   #-3445   (54)   #-3464   (54)   #-3785   (57)
                                                            #-3887   (58)   #-3989   (59)   #-4242   (61)   #-4287   (61)
                                                            #-4323   (61)   #-4450   (62)   #-4485   (62)   #-4603   (63)
                                                            #-4739   (64)   #-4774   (64)   #-4892   (65)   #-5026   (66)
                                                            #-5060   (66)   #-5171   (67)   #-5365   (68)   #-5400   (68)
                                                            #-5449   (68)   #-5644   (69)   #-5678   (69)   #-5725   (69)
                                                            #-5911   (70)   #-5946   (70)   #-5993   (70)   #-6186   (71)
                                                            #-6220   (71)   #-6264   (71)   #-6409   (72)   #-6452   (72)
                                                            #-6573   (73)   #-6584   (73)   #-6684   (74)   #-6699   (74)
                                                            #-6717   (74)   #-6874   (77)   #-6976   (78)   #-7078   (79)
                                                            #-7331   (81)   #-7376   (81)   #-7412   (81)   #-7539   (82)
                                                            #-7574   (82)   #-7692   (83)   #-7828   (84)   #-7863   (84)
                                                            #-7981   (85)   #-8115   (86)   #-8149   (86)   #-8260   (87)
```

```
                                                              #-8454   (88)    #-8489   (88)    #-8538   (88)    #-8733   (89)
                                                              #-8767   (89)    #-8814   (89)    #-9000   (90)    #-9035   (90)
                                                              #-9082   (90)    #-9275   (91)    #-9309   (91)    #-9353   (91)
                                                              #-9498   (92)    #-9541   (92)    #-9661   (93)    #-9672   (93)
                                                              #-9772   (94)    #-9787   (94)    #-9806   (94)
V_DATA_FLAG                      =00000002      175     (2)   #-3291   (53)
V_DONE                           =00000007      137     (2)   #-3446   (54)    #-3465   (54)    #-3542   (54)    #-3574   (54)
V_DONE_FLAG                      =00000000      185     (2)
V_END_FLAG                       =00000003      177     (2)   #-3295   (53)
V_RD_INT_FLAG                    =00000002      189     (2)   #-6700   (74)    #-6716   (74)    #-9788   (94)    #-9805   (94)
V_READY                          =00000007      138     (2)   #-3423   (54)    #-3551   (54)    #-3581   (54)
V_RECEIVE_FLAG                   =00000001      173     (2)   #-3692   (56)    #-4083   (60)    #-6781   (76)    #-7172   (80)
V_REPEAT_CHECK                   =00000002      183     (2)   #-6002   (70)    #-6030   (70)    #-6273   (71)    #-6298   (71)
                                                              #-9091   (90)    #-9119   (90)    #-9362   (91)    #-9387   (91)
V_REPEAT_SEND                    =00000001      181     (2)   #-3686   (56)    #-3693   (56)    #-3704   (56)    #-4077   (60)
                                                              #-4084   (60)    #-4095   (60)    #-5954   (70)    #-6227   (71)
                                                              #-6775   (76)    #-6782   (76)    #-6793   (76)    #-7166   (80)
                                                              #-7173   (80)    #-7184   (80)    #-9043   (90)    #-9316   (91)
V_RESPONSE_FLAG                  =00000004      146     (2)
V_RUN                            =00000000      142     (2)
V_SDONE_FLAG                     =00000001      187     (2)   #-6451   (72)    #-9540   (92)
V_SEND_FLAG                      =00000004      179     (2)   #-3685   (56)    #-4076   (60)    #-6774   (76)    #-7165   (80)
V_TRANSFER                       =00000004      139     (2)
WCS                              =00000007      94      (2)
WRITE                            =00000003      115     (2)     1030   (18)      1064   (18)      1098   (19)      1131   (20)
                                                                1165   (20)      1320   (23)      1337   (23)      1370   (24)
                                                                1403   (25)      1437   (25)      1454   (26)      1471   (26)
                                                                1522   (27)      1539   (27)      1556   (28)      1607   (29)
                                                                1641   (29)      1675   (30)      1709   (31)      1743   (31)
                                                                1777   (32)      1810   (33)      1844   (33)      1877   (34)
                                                                1910   (35)      1944   (35)      1978   (36)      2011   (37)
                                                                2045   (37)      440    (6)       457    (6)       490    (7)
                                                                523    (8)       557    (8)       574    (9)       591    (9)
                                                                642    (10)      659    (10)      676    (11)      727    (12)
                                                                761    (12)      795    (13)      829    (14)      863    (14)
                                                                897    (15)      930    (16)      964    (16)      997    (17)
WRITE_SERVICE                     000021D4-R    3591    (54)    6397   (72)      9486   (92)
XOFF_FLAG                        =00000013      108     (2)
XON_FLAG                         =00000011      107     (2)
```

```
                                              +------------------------+
                                              ! Macros Cross Reference !
                                              +------------------------+

MACRO                  SIZE             DEFINITION            REFERENCES...
-----                  ----             ----------            -------------
$D1_ABPROGRAM          1                3690    (56)          3690    (56)          3698    (56)          3711    (56)          3789    (57)          3804    (57)
                                                              3812    (57)          3891    (58)          3906    (58)          3914    (58)          3993    (59)
                                                              4008    (59)          4016    (59)          4081    (60)          4089    (60)          4102    (60)
                                                              4246    (61)          4258    (61)          4269    (61)          4291    (61)          4308    (61)
                                                              4317    (61)          4327    (61)          4345    (61)          4353    (61)          4454    (62)
                                                              4470    (62)          4479    (62)          4489    (62)          4508    (62)          4517    (62)
                                                              4607    (63)          4623    (63)          4633    (63)          4645    (63)          4743    (64)
                                                              4759    (64)          4768    (64)          4778    (64)          4796    (64)          4805    (64)
                                                              4896    (65)          4912    (65)          4922    (65)          4934    (65)          5030    (66)
                                                              5045    (66)          5054    (66)          5064    (66)          5081    (66)          5090    (66)
                                                              5175    (67)          5191    (67)          5200    (67)          5211    (67)          5369    (68)
                                                              5385    (68)          5394    (68)          5404    (68)          5423    (68)          5432    (68)
                                                              5453    (68)          5469    (68)          5479    (68)          5491    (68)          5648    (69)
                                                              5663    (69)          5672    (69)          5682    (69)          5700    (69)          5709    (69)
                                                              5729    (69)          5745    (69)          5754    (69)          5765    (69)          5915    (70)
                                                              5931    (70)          5940    (70)          5950    (70)          5973    (70)          5982    (70)
                                                              5997    (70)          6016    (70)          6026    (70)          6042    (70)          6190    (71)
                                                              6205    (71)          6214    (71)          6224    (71)          6245    (71)          6253    (71)
                                                              6268    (71)          6287    (71)          6296    (71)          6311    (71)          6413    (72)
                                                              6428    (72)          6437    (72)          6459    (72)          6476    (72)          6485    (72)
                                                              6577    (73)          6596    (73)          6604    (73)          6614    (73)          6688    (74)
                                                              6705    (74)          6722    (74)          6779    (76)          6787    (76)          6800    (76)
                                                              6878    (77)          6893    (77)          6901    (77)          6980    (78)          6995    (78)
                                                              7003    (78)          7082    (79)          7097    (79)          7105    (79)          7170    (80)
                                                              7178    (80)          7191    (80)          7335    (81)          7347    (81)          7358    (81)
                                                              7380    (81)          7397    (81)          7406    (81)          7416    (81)          7434    (81)
                                                              7442    (81)          7543    (82)          7559    (82)          7568    (82)          7578    (82)
                                                              7597    (82)          7606    (82)          7696    (83)          7712    (83)          7722    (83)
                                                              7734    (83)          7832    (84)          7848    (84)          7857    (84)          7867    (84)
                                                              7885    (84)          7894    (84)          7985    (85)          8001    (85)          8011    (85)
                                                              8023    (85)          8119    (86)          8134    (86)          8143    (86)          8153    (86)
                                                              8170    (86)          8179    (86)          8264    (87)          8280    (87)          8289    (87)
                                                              8300    (87)          8458    (88)          8474    (88)          8483    (88)          8493    (88)
                                                              8512    (88)          8521    (88)          8542    (88)          8558    (88)          8568    (88)
                                                              8580    (88)          8737    (89)          8752    (89)          8761    (89)          8771    (89)
                                                              8789    (89)          8798    (89)          8818    (89)          8834    (89)          8843    (89)
                                                              8854    (89)          9004    (90)          9020    (90)          9029    (90)          9039    (90)
                                                              9062    (90)          9071    (90)          9086    (90)          9105    (90)          9115    (90)
                                                              9131    (90)          9279    (91)          9294    (91)          9303    (91)          9313    (91)
                                                              9334    (91)          9342    (91)          9357    (91)          9376    (91)          9385    (91)
                                                              9400    (91)          9502    (92)          9517    (92)          9526    (92)          9548    (92)
                                                              9565    (92)          9574    (92)          9665    (93)          9684    (93)          9692    (93)
                                                              9702    (93)          9776    (94)          9793    (94)          9811    (94)
$D1 ABTEST             1                3689    (56)          3689    (56)          3697    (56)          3710    (56)          3788    (57)          3803    (57)
                                                              3890    (58)          3905    (58)          3992    (59)          4007    (59)          4080    (60)
                                                              4088    (60)          4101    (60)          4245    (61)          4256    (61)          4290    (61)
                                                              4306    (61)          4316    (61)          4326    (61)          4343    (61)          4453    (62)
                                                              4468    (62)          4478    (62)          4488    (62)          4506    (62)          4516    (62)
```

```
                                                   4606   (63)      4621   (63)      4632   (63)      4644   (63)      4742   (64)
                                                   4757   (64)      4767   (64)      4777   (64)      4794   (64)      4804   (64)
                                                   4895   (65)      4910   (65)      4921   (65)      4933   (65)      5029   (66)
                                                   5043   (66)      5053   (66)      5063   (66)      5079   (66)      5089   (66)
                                                   5174   (67)      5189   (67)      5199   (67)      5210   (67)      5368   (68)
                                                   5383   (68)      5393   (68)      5403   (68)      5421   (68)      5431   (68)
                                                   5452   (68)      5467   (68)      5478   (68)      5490   (68)      5647   (69)
                                                   5661   (69)      5671   (69)      5681   (69)      5698   (69)      5708   (69)
                                                   5728   (69)      5743   (69)      5753   (69)      5764   (69)      5914   (70)
                                                   5929   (70)      5939   (70)      5949   (70)      5971   (70)      5981   (70)
                                                   5996   (70)      6014   (70)      6025   (70)      6041   (70)      6189   (71)
                                                   6203   (71)      6213   (71)      6223   (71)      6243   (71)      6267   (71)
                                                   6285   (71)      6295   (71)      6310   (71)      6412   (72)      6426   (72)
                                                   6436   (72)      6458   (72)      6474   (72)      6484   (72)      6576   (73)
                                                   6595   (73)      6603   (73)      6613   (73)      6687   (74)      6704   (74)
                                                   6721   (74)      6778   (76)      6786   (76)      6799   (76)      6877   (77)
                                                   6892   (77)      6979   (78)      6994   (78)      7081   (79)      7096   (79)
                                                   7169   (80)      7177   (80)      7190   (80)      7334   (81)      7345   (81)
                                                   7379   (81)      7395   (81)      7405   (81)      7415   (81)      7432   (81)
                                                   7542   (82)      7557   (82)      7567   (82)      7577   (82)      7595   (82)
                                                   7605   (82)      7695   (83)      7710   (83)      7721   (83)      7733   (83)
                                                   7831   (84)      7846   (84)      7856   (84)      7866   (84)      7883   (84)
                                                   7893   (84)      7984   (85)      7999   (85)      8010   (85)      8022   (85)
                                                   8118   (86)      8132   (86)      8142   (86)      8152   (86)      8168   (86)
                                                   8178   (86)      8263   (87)      8278   (87)      8288   (87)      8299   (87)
                                                   8457   (88)      8472   (88)      8482   (88)      8492   (88)      8510   (88)
                                                   8520   (88)      8541   (88)      8556   (88)      8567   (88)      8579   (88)
                                                   8736   (89)      8750   (89)      8760   (89)      8770   (89)      8787   (89)
                                                   8797   (89)      8817   (89)      8832   (89)      8842   (89)      8853   (89)
                                                   9003   (90)      9018   (90)      9028   (90)      9038   (90)      9060   (90)
                                                   9070   (90)      9085   (90)      9103   (90)      9114   (90)      9130   (90)
                                                   9278   (91)      9292   (91)      9302   (91)      9312   (91)      9332   (91)
                                                   9356   (91)      9374   (91)      9384   (91)      9399   (91)      9501   (92)
                                                   9515   (92)      9525   (92)      9547   (92)      9563   (92)      9573   (92)
                                                   9664   (93)      9683   (93)      9691   (93)      9701   (93)      9775   (94)
                                                   9792   (94)      9810   (94)
$D1_BSUBT              1              3775   (57)  3775   (57)      3876   (58)      3978   (59)      4231   (61)      4439   (62)
                                                   4592   (63)      4728   (64)      4881   (65)      5014   (66)      5160   (67)
                                                   5350   (68)      5629   (69)      5896   (70)      6171   (71)      6396   (72)
                                                   6559   (73)      6674   (74)      6864   (77)      6965   (78)      7067   (79)
                                                   7320   (81)      7528   (82)      7681   (83)      7817   (84)      7970   (85)
                                                   8103   (86)      8249   (87)      8439   (88)      8718   (89)      8985   (90)
                                                   9260   (91)      9485   (92)      9648   (93)      9762   (94)
$D1_BTEST              1              3668   (56)  3668   (56)      4059   (60)      6757   (76)      7148   (80)
$D1_ERR                1              3687   (56)  3687   (56)      3695   (56)      3708   (56)      3786   (57)      3801   (57)
                                                   3810   (57)      3888   (58)      3903   (58)      3912   (58)      3990   (59)
                                                   4005   (59)      4014   (59)      4078   (60)      4086   (60)      4099   (60)
                                                   4243   (61)      4254   (61)      4267   (61)      4288   (61)      4304   (61)
                                                   4314   (61)      4324   (61)      4341   (61)      4351   (61)      4451   (62)
                                                   4466   (62)      4476   (62)      4486   (62)      4504   (62)      4514   (62)
                                                   4604   (63)      4619   (63)      4630   (63)      4642   (63)      4740   (64)
                                                   4755   (64)      4765   (64)      4775   (64)      4792   (64)      4802   (64)
                                                   4893   (65)      4908   (65)      4919   (65)      4931   (65)      5027   (66)
                                                   5041   (66)      5051   (66)      5061   (66)      5077   (66)      5087   (66)
                                                   5172   (67)      5187   (67)      5197   (67)      5208   (67)      5366   (68)
```

```
                                     5381  (68)   5391  (68)   5401  (68)   5419  (68)   5429  (68)
                                     5450  (68)   5465  (68)   5476  (68)   5488  (68)   5645  (69)
                                     5659  (69)   5669  (69)   5679  (69)   5696  (69)   5706  (69)
                                     5726  (69)   5741  (69)   5751  (69)   5762  (69)   5912  (70)
                                     5927  (70)   5937  (70)   5947  (70)   5969  (70)   5979  (70)
                                     5994  (70)   6012  (70)   6023  (70)   6039  (70)   6187  (71)
                                     6201  (71)   6211  (71)   6221  (71)   6241  (71)   6251  (71)
                                     6265  (71)   6283  (71)   6293  (71)   6308  (71)   6410  (72)
                                     6424  (72)   6434  (72)   6456  (72)   6472  (72)   6482  (72)
                                     6574  (73)   6593  (73)   6601  (73)   6611  (73)   6685  (74)
                                     6702  (74)   6719  (74)   6776  (76)   6784  (76)   6797  (76)
                                     6875  (77)   6890  (77)   6899  (77)   6977  (78)   6992  (78)
                                     7001  (78)   7079  (79)   7094  (79)   7103  (79)   7167  (80)
                                     7175  (80)   7188  (80)   7332  (81)   7343  (81)   7356  (81)
                                     7377  (81)   7393  (81)   7403  (81)   7413  (81)   7430  (81)
                                     7440  (81)   7540  (82)   7555  (82)   7565  (82)   7575  (82)
                                     7593  (82)   7603  (82)   7693  (83)   7708  (83)   7719  (83)
                                     7731  (83)   7829  (84)   7844  (84)   7854  (84)   7864  (84)
                                     7881  (84)   7891  (84)   7982  (85)   7997  (85)   8008  (85)
                                     8020  (85)   8116  (86)   8130  (86)   8140  (86)   8150  (86)
                                     8166  (86)   8176  (86)   8261  (87)   8276  (87)   8286  (87)
                                     8297  (87)   8455  (88)   8470  (88)   8480  (88)   8490  (88)
                                     8508  (88)   8518  (88)   8539  (88)   8554  (88)   8565  (88)
                                     8577  (88)   8734  (89)   8748  (89)   8758  (89)   8768  (89)
                                     8785  (89)   8795  (89)   8815  (89)   8830  (89)   8840  (89)
                                     8851  (89)   9001  (90)   9016  (90)   9026  (90)   9036  (90)
                                     9058  (90)   9068  (90)   9083  (90)   9101  (90)   9112  (90)
                                     9128  (90)   9276  (91)   9290  (91)   9300  (91)   9310  (91)
                                     9330  (91)   9340  (91)   9354  (91)   9372  (91)   9382  (91)
                                     9397  (91)   9499  (92)   9513  (92)   9523  (92)   9545  (92)
                                     9561  (92)   9571  (92)   9662  (93)   9681  (93)   9689  (93)
                                     9699  (93)   9773  (94)   9790  (94)   9808  (94)
$D1_ERR_S     1         3687  (56)   3687  (56)   3695  (56)   3708  (56)   3786  (57)   3801  (57)
                                     3810  (57)   3888  (58)   3903  (58)   3912  (58)   3990  (59)
                                     4005  (59)   4014  (59)   4078  (60)   4086  (60)   4099  (60)
                                     4243  (61)   4254  (61)   4267  (61)   4288  (61)   4304  (61)
                                     4314  (61)   4324  (61)   4341  (61)   4351  (61)   4451  (62)
                                     4466  (62)   4476  (62)   4486  (62)   4504  (62)   4514  (62)
                                     4604  (63)   4619  (63)   4630  (63)   4642  (63)   4740  (64)
                                     4755  (64)   4765  (64)   4775  (64)   4792  (64)   4802  (64)
                                     4893  (65)   4908  (65)   4919  (65)   4931  (65)   5027  (66)
                                     5041  (66)   5051  (66)   5061  (66)   5077  (66)   5087  (66)
                                     5172  (67)   5187  (67)   5197  (67)   5208  (67)   5366  (68)
                                     5381  (68)   5391  (68)   5401  (68)   5419  (68)   5429  (68)
                                     5450  (68)   5465  (68)   5476  (68)   5488  (68)   5645  (69)
                                     5659  (69)   5669  (69)   5679  (69)   5696  (69)   5706  (69)
                                     5726  (69)   5741  (69)   5751  (69)   5762  (69)   5912  (70)
                                     5927  (70)   5937  (70)   5947  (70)   5969  (70)   5979  (70)
                                     5994  (70)   6012  (70)   6023  (70)   6039  (70)   6187  (71)
                                     6201  (71)   6211  (71)   6221  (71)   6241  (71)   6251  (71)
                                     6265  (71)   6283  (71)   6293  (71)   6308  (71)   6410  (72)
                                     6424  (72)   6434  (72)   6456  (72)   6472  (72)   6482  (72)
                                     6574  (73)   6593  (73)   6601  (73)   6611  (73)   6685  (74)
                                     6702  (74)   6719  (74)   6776  (76)   6784  (76)   6797  (76)
                                     6875  (77)   6890  (77)   6899  (77)   6977  (78)   6992  (78)
```

```
                                                            7001   (78)       7079   (79)       7094   (79)       7103   (79)       7167   (80)
                                                            7175   (80)       7188   (80)       7332   (81)       7343   (81)       7356   (81)
                                                            7377   (81)       7393   (81)       7403   (81)       7413   (81)       7430   (81)
                                                            7440   (81)       7540   (82)       7555   (82)       7565   (82)       7575   (82)
                                                            7593   (82)       7603   (82)       7693   (83)       7708   (83)       7719   (83)
                                                            7731   (83)       7829   (84)       7844   (84)       7854   (84)       7864   (84)
                                                            7881   (84)       7891   (84)       7982   (85)       7997   (85)       8008   (85)
                                                            8020   (85)       8116   (86)       8130   (86)       8140   (86)       8150   (86)
                                                            8166   (86)       8176   (86)       8261   (87)       8276   (87)       8286   (87)
                                                            8297   (87)       8455   (88)       8470   (88)       8480   (88)       8490   (88)
                                                            8508   (88)       8518   (88)       8539   (88)       8554   (88)       8565   (88)
                                                            8577   (88)       8734   (89)       8748   (89)       8758   (89)       8768   (89)
                                                            8785   (89)       8795   (89)       8815   (89)       8830   (89)       8840   (89)
                                                            8851   (89)       9001   (90)       9016   (90)       9026   (90)       9036   (90)
                                                            9058   (90)       9068   (90)       9083   (90)       9101   (90)       9112   (90)
                                                            9128   (90)       9276   (91)       9290   (91)       9300   (91)       9310   (91)
                                                            9330   (91)       9340   (91)       9354   (91)       9372   (91)       9382   (91)
                                                            9397   (91)       9499   (92)       9513   (92)       9523   (92)       9545   (92)
                                                            9561   (92)       9571   (92)       9662   (93)       9681   (93)       9689   (93)
                                                            9699   (93)       9773   (94)       9790   (94)       9808   (94)
$D1_ESUBT               1                 3815    (57)      3815   (57)       3917   (58)       4019   (59)       4358   (61)       4519   (62)
                                                            4648   (63)       4807   (64)       4935   (65)       5092   (66)       5213   (67)
                                                            5497   (68)       5771   (69)       6046   (70)       6315   (71)       6487   (72)
                                                            6616   (73)       6727   (74)       6904   (77)       7006   (78)       7108   (79)
                                                            7447   (81)       7608   (82)       7737   (83)       7896   (84)       8024   (85)
                                                            8181   (86)       8302   (87)       8586   (88)       8860   (89)       9135   (90)
                                                            9404   (91)       9576   (92)       9704   (93)       9816   (94)
$D1 ETEST               1                 4022    (59)      4022   (59)       6733   (74)       7111   (79)       9822   (94)
$D1 EXSUB               1                 4279    (61)      4279   (61)       7368   (81)
$D1 EXTEST              1                 4281    (61)      4281   (61)       7370   (81)
$D1 PRINT_S             1                 3173    (53)      3173   (53)       3178   (53)       3183   (53)       3200   (53)       3262   (53)
                                                            3263   (53)       3264   (53)       3269   (53)       3270   (53)       3271   (53)
                                                            3276   (53)       3290   (53)       3292   (53)       3294   (53)       3300   (53)
                                                            3306   (53)       3309   (53)       3310   (53)       3316   (53)       3321   (53)
                                                            3610   (54)       3614   (54)       3616   (54)       3626   (54)
$D1 SBTTL               2                 3649    (56)      3649   (56)       3715   (57)       3819   (58)       3921   (59)       4026   (60)
                                                            4106   (61)       4362   (62)       4523   (63)       4652   (64)       4811   (65)
                                                            4938   (66)       5096   (67)       5217   (68)       5501   (69)       5775   (70)
                                                            6050   (71)       6318   (72)       6490   (73)       6619   (74)       6738   (76)
                                                            6804   (77)       6908   (78)       7010   (79)       7115   (80)       7195   (81)
                                                            7451   (82)       7612   (83)       7741   (84)       7900   (85)       8027   (86)
                                                            8185   (87)       8306   (88)       8590   (89)       8864   (90)       9139   (91)
                                                            9407   (92)       9579   (93)       9707   (94)
$D2_BDATA               1                 3668    (56)      3668   (56)       4059   (60)       6757   (76)       7148   (80)
$D2_BTEST               1                 3668    (56)      3668   (56)       4059   (60)       6757   (76)       7148   (80)
$D2_SBTTL               1                 3649    (56)      3649   (56)       4026   (60)       6738   (76)       7115   (80)
$DEF                    1                 92      (2)
$DEFINI                 1                 88      (2)       88     (2)        91     (2)        93     (2)
$DS_ABORT               1                 3689    (56)      3689   (56)       3690   (56)       3697   (56)       3698   (56)       3710   (56)
                                                            3711   (56)       3788   (57)       3789   (57)       3803   (57)       3804   (57)
                                                            3812   (57)       3890   (58)       3891   (58)       3905   (58)       3906   (58)
                                                            3914   (58)       3992   (59)       3993   (59)       4007   (59)       4008   (59)
                                                            4016   (59)       4080   (60)       4081   (60)       4088   (60)       4089   (60)
                                                            4101   (60)       4102   (60)       4245   (61)       4246   (61)       4256   (61)
                                                            4258   (61)       4269   (61)       4290   (61)       4291   (61)       4306   (61)
```

```
4308   (61)     4316   (61)     4317   (61)     4326   (61)     4327   (61)
4343   (61)     4345   (61)     4353   (61)     4453   (62)     4454   (62)
4468   (62)     4470   (62)     4478   (62)     4479   (62)     4488   (62)
4489   (62)     4506   (62)     4508   (62)     4516   (62)     4517   (62)
4606   (63)     4607   (63)     4621   (63)     4623   (63)     4632   (63)
4633   (63)     4644   (63)     4645   (63)     4742   (64)     4743   (64)
4757   (64)     4759   (64)     4767   (64)     4768   (64)     4777   (64)
4778   (64)     4794   (64)     4796   (64)     4804   (64)     4805   (64)
4895   (65)     4896   (65)     4910   (65)     4912   (65)     4921   (65)
4922   (65)     4933   (65)     4934   (65)     5029   (66)     5030   (66)
5043   (66)     5045   (66)     5053   (66)     5054   (66)     5063   (66)
5064   (66)     5079   (66)     5081   (66)     5089   (66)     5090   (66)
5174   (67)     5175   (67)     5189   (67)     5191   (67)     5199   (67)
5200   (67)     5210   (67)     5211   (67)     5368   (68)     5369   (68)
5383   (68)     5385   (68)     5393   (68)     5394   (68)     5403   (68)
5404   (68)     5421   (68)     5423   (68)     5431   (68)     5432   (68)
5452   (68)     5453   (68)     5467   (68)     5469   (68)     5478   (68)
5479   (68)     5490   (68)     5491   (68)     5647   (69)     5648   (69)
5661   (69)     5663   (69)     5671   (69)     5672   (69)     5681   (69)
5682   (69)     5698   (69)     5700   (69)     5708   (69)     5709   (69)
5728   (69)     5729   (69)     5743   (69)     5745   (69)     5753   (69)
5754   (69)     5764   (69)     5765   (69)     5914   (70)     5915   (70)
5929   (70)     5931   (70)     5939   (70)     5940   (70)     5949   (70)
5950   (70)     5971   (70)     5973   (70)     5981   (70)     5982   (70)
5996   (70)     5997   (70)     6014   (70)     6016   (70)     6025   (70)
6026   (70)     6041   (70)     6042   (70)     6189   (71)     6190   (71)
6203   (71)     6205   (71)     6213   (71)     6214   (71)     6223   (71)
6224   (71)     6243   (71)     6245   (71)     6253   (71)     6267   (71)
6268   (71)     6285   (71)     6287   (71)     6295   (71)     6296   (71)
6310   (71)     6311   (71)     6412   (72)     6413   (72)     6426   (72)
6428   (72)     6436   (72)     6437   (72)     6458   (72)     6459   (72)
6474   (72)     6476   (72)     6484   (72)     6485   (72)     6576   (73)
6577   (73)     6595   (73)     6596   (73)     6603   (73)     6604   (73)
6613   (73)     6614   (73)     6687   (74)     6688   (74)     6704   (74)
6705   (74)     6721   (74)     6722   (74)     6778   (76)     6779   (76)
6786   (76)     6787   (76)     6799   (76)     6800   (76)     6877   (77)
6878   (77)     6892   (77)     6893   (77)     6901   (77)     6979   (78)
6980   (78)     6994   (78)     6995   (78)     7003   (78)     7081   (79)
7082   (79)     7096   (79)     7097   (79)     7105   (79)     7169   (80)
7170   (80)     7177   (80)     7178   (80)     7190   (80)     7191   (80)
7334   (81)     7335   (81)     7345   (81)     7347   (81)     7358   (81)
7379   (81)     7380   (81)     7395   (81)     7397   (81)     7405   (81)
7406   (81)     7415   (81)     7416   (81)     7432   (81)     7434   (81)
7442   (81)     7542   (82)     7543   (82)     7557   (82)     7559   (82)
7567   (82)     7568   (82)     7577   (82)     7578   (82)     7595   (82)
7597   (82)     7605   (82)     7606   (82)     7695   (83)     7696   (83)
7710   (83)     7712   (83)     7721   (83)     7722   (83)     7733   (83)
7734   (83)     7831   (84)     7832   (84)     7846   (84)     7848   (84)
7856   (84)     7857   (84)     7866   (84)     7867   (84)     7883   (84)
7885   (84)     7893   (84)     7894   (84)     7984   (85)     7985   (85)
7999   (85)     8001   (85)     8010   (85)     8011   (85)     8022   (85)
8023   (85)     8118   (86)     8119   (86)     8132   (86)     8134   (86)
8142   (86)     8143   (86)     8152   (86)     8153   (86)     8168   (86)
8170   (86)     8178   (86)     8179   (86)     8263   (87)     8264   (87)
8278   (87)     8280   (87)     8288   (87)     8289   (87)     8299   (87)
```

```
                                                    8300   (87)      8457   (88)      8458   (88)      8472   (88)      8474   (88)
                                                    8482   (88)      8483   (88)      8492   (88)      8493   (88)      8510   (88)
                                                    8512   (88)      8520   (88)      8521   (88)      8541   (88)      8542   (88)
                                                    8556   (88)      8558   (88)      8567   (88)      8568   (88)      8579   (88)
                                                    8580   (88)      8736   (89)      8737   (89)      8750   (89)      8752   (89)
                                                    8760   (89)      8761   (89)      8770   (89)      8771   (89)      8787   (89)
                                                    8789   (89)      8797   (89)      8798   (89)      8817   (89)      8818   (89)
                                                    8832   (89)      8834   (89)      8842   (89)      8843   (89)      8853   (89)
                                                    8854   (89)      9003   (90)      9004   (90)      9018   (90)      9020   (90)
                                                    9028   (90)      9029   (90)      9038   (90)      9039   (90)      9060   (90)
                                                    9062   (90)      9070   (90)      9071   (90)      9085   (90)      9086   (90)
                                                    9103   (90)      9105   (90)      9114   (90)      9115   (90)      9130   (90)
                                                    9131   (90)      9278   (91)      9279   (91)      9292   (91)      9294   (91)
                                                    9302   (91)      9303   (91)      9312   (91)      9313   (91)      9332   (91)
                                                    9334   (91)      9342   (91)      9356   (91)      9357   (91)      9374   (91)
                                                    9376   (91)      9384   (91)      9385   (91)      9399   (91)      9400   (91)
                                                    9501   (92)      9502   (92)      9515   (92)      9517   (92)      9525   (92)
                                                    9526   (92)      9547   (92)      9548   (92)      9563   (92)      9565   (92)
                                                    9573   (92)      9574   (92)      9664   (93)      9665   (93)      9683   (93)
                                                    9684   (93)      9691   (93)      9692   (93)      9701   (93)      9702   (93)
                                                    9775   (94)      9776   (94)      9792   (94)      9793   (94)      9810   (94)
                                                    9811   (94)
$DS_ASKLGCL_S       1          4280     (61)        4280   (61)      7369   (81)
$DS_BCOMPLETE       1          3811     (57)        3811   (57)      3913   (58)      4015   (59)      4268   (61)      4278   (61)
                                                    4352   (61)      4643   (63)      6252   (71)      6900   (77)      7002   (78)
                                                    7104   (79)      7357   (81)      7367   (81)      7441   (81)      7732   (83)
                                                    9341   (91)
$DS BGNMESSAGE      1          3172     (53)        3172   (53)      3177   (53)      3182   (53)      3199   (53)      3261   (53)
                                                    3268   (53)      3275   (53)      3289   (53)      3315   (53)      3320   (53)
$DS_BGNMOD          1          72       (2)         72     (2)
$DS_BGNSUB          1          3775     (57)        3775   (57)      3876   (58)      3978   (59)      4231   (61)      4439   (62)
                                                    4592   (63)      4728   (64)      4881   (65)      5014   (66)      5160   (67)
                                                    5350   (68)      5629   (69)      5896   (70)      6171   (71)      6396   (72)
                                                    6559   (73)      6674   (74)      6864   (77)      6965   (78)      7067   (79)
                                                    7320   (81)      7528   (82)      7681   (83)      7817   (84)      7970   (85)
                                                    8103   (86)      8249   (87)      8439   (88)      8718   (89)      8985   (90)
                                                    9260   (91)      9485   (92)      9648   (93)      9762   (94)
$DS_BGNTEST         1          3668     (56)        3668   (56)      4059   (60)      6757   (76)      7148   (80)
$DS_BNCOMPLETE      1          3305     (53)        3305   (53)      3688   (56)      3696   (56)      3703   (56)      3709   (56)
                                                    3787   (57)      3802   (57)      3809   (57)      3889   (58)      3904   (58)
                                                    3911   (58)      3991   (59)      4006   (59)      4013   (59)      4079   (60)
                                                    4087   (60)      4094   (60)      4100   (60)      4244   (61)      4255   (61)
                                                    4266   (61)      4289   (61)      4305   (61)      4313   (61)      4315   (61)
                                                    4325   (61)      4342   (61)      4350   (61)      4452   (62)      4467   (62)
                                                    4475   (62)      4477   (62)      4487   (62)      4505   (62)      4513   (62)
                                                    4515   (62)      4605   (63)      4620   (63)      4629   (63)      4631   (63)
                                                    4641   (63)      4741   (64)      4756   (64)      4764   (64)      4766   (64)
                                                    4776   (64)      4793   (64)      4801   (64)      4803   (64)      4894   (65)
                                                    4909   (65)      4918   (65)      4920   (65)      4930   (65)      4932   (65)
                                                    5028   (66)      5042   (66)      5050   (66)      5052   (66)      5062   (66)
                                                    5078   (66)      5086   (66)      5088   (66)      5173   (67)      5188   (67)
                                                    5196   (67)      5198   (67)      5207   (67)      5209   (67)      5367   (68)
                                                    5382   (68)      5390   (68)      5392   (68)      5402   (68)      5420   (68)
                                                    5428   (68)      5430   (68)      5451   (68)      5466   (68)      5475   (68)
                                                    5477   (68)      5487   (68)      5489   (68)      5646   (69)      5660   (69)
                                                    5668  (69)      5670  (69)      5680  (69)      5697  (69)      5705  (69)
                                                    5707  (69)      5727  (69)      5742  (69)      5750  (69)      5752  (69)
                                                    5761  (69)      5763  (69)      5913  (70)      5928  (70)      5936  (70)
                                                    5938  (70)      5948  (70)      5970  (70)      5978  (70)      5980  (70)
                                                    5995  (70)      6013  (70)      6022  (70)      6024  (70)      6038  (70)
                                                    6040  (70)      6188  (71)      6202  (71)      6210  (71)      6212  (71)
                                                    6222  (71)      6242  (71)      6250  (71)      6266  (71)      6284  (71)
                                                    6292  (71)      6294  (71)      6307  (71)      6309  (71)      6411  (72)
                                                    6425  (72)      6433  (72)      6435  (72)      6457  (72)      6473  (72)
                                                    6481  (72)      6483  (72)      6575  (73)      6594  (73)      6600  (73)
                                                    6602  (73)      6610  (73)      6612  (73)      6686  (74)      6703  (74)
                                                    6720  (74)      6777  (76)      6785  (76)      6792  (76)      6798  (76)
                                                    6876  (77)      6891  (77)      6898  (77)      6978  (78)      6993  (78)
                                                    7000  (78)      7080  (79)      7095  (79)      7102  (79)      7168  (80)
                                                    7176  (80)      7183  (80)      7189  (80)      7333  (81)      7344  (81)
                                                    7355  (81)      7378  (81)      7394  (81)      7402  (81)      7404  (81)
                                                    7414  (81)      7431  (81)      7439  (81)      7541  (82)      7556  (82)
                                                    7564  (82)      7566  (82)      7576  (82)      7594  (82)      7602  (82)
                                                    7604  (82)      7694  (83)      7709  (83)      7718  (83)      7720  (83)
                                                    7730  (83)      7830  (84)      7845  (84)      7853  (84)      7855  (84)
                                                    7865  (84)      7882  (84)      7890  (84)      7892  (84)      7983  (85)
                                                    7998  (85)      8007  (85)      8009  (85)      8019  (85)      8021  (85)
                                                    8117  (86)      8131  (86)      8139  (86)      8141  (86)      8151  (86)
                                                    8167  (86)      8175  (86)      8177  (86)      8262  (87)      8277  (87)
                                                    8285  (87)      8287  (87)      8296  (87)      8298  (87)      8456  (88)
                                                    8471  (88)      8479  (88)      8481  (88)      8491  (88)      8509  (88)
                                                    8517  (88)      8519  (88)      8540  (88)      8555  (88)      8564  (88)
                                                    8566  (88)      8576  (88)      8578  (88)      8735  (89)      8749  (89)
                                                    8757  (89)      8759  (89)      8769  (89)      8786  (89)      8794  (89)
                                                    8796  (89)      8816  (89)      8831  (89)      8839  (89)      8841  (89)
                                                    8850  (89)      8852  (89)      9002  (90)      9017  (90)      9025  (90)
                                                    9027  (90)      9037  (90)      9059  (90)      9067  (90)      9069  (90)
                                                    9084  (90)      9102  (90)      9111  (90)      9113  (90)      9127  (90)
                                                    9129  (90)      9277  (91)      9291  (91)      9299  (91)      9301  (91)
                                                    9311  (91)      9331  (91)      9339  (91)      9355  (91)      9373  (91)
                                                    9381  (91)      9383  (91)      9396  (91)      9398  (91)      9500  (92)
                                                    9514  (92)      9522  (92)      9524  (92)      9546  (92)      9562  (92)
                                                    9570  (92)      9572  (92)      9663  (93)      9682  (93)      9688  (93)
                                                    9690  (93)      9698  (93)      9700  (93)      9774  (94)      9791  (94)
                                                    9809  (94)
$DS_BREAK               1                3701  (56) 3701  (56)      3807  (57)      3909  (58)      4011  (59)      4022  (59)
                                                    4092  (60)      4264  (61)      4276  (61)      4311  (61)      4348  (61)
                                                    4473  (62)      4511  (62)      4627  (63)      4639  (63)      4762  (64)
                                                    4799  (64)      4916  (65)      4928  (65)      5048  (66)      5084  (66)
                                                    5194  (67)      5205  (67)      5388  (68)      5426  (68)      5473  (68)
                                                    5485  (68)      5666  (69)      5703  (69)      5748  (69)      5759  (69)
                                                    5934  (70)      5976  (70)      6020  (70)      6036  (70)      6208  (71)
                                                    6248  (71)      6290  (71)      6305  (71)      6431  (72)      6450  (72)
                                                    6479  (72)      6583  (73)      6598  (73)      6608  (73)      6698  (74)
                                                    6715  (74)      6733  (74)      6790  (76)      6896  (77)      6998  (78)
                                                    7100  (79)      7111  (79)      7181  (80)      7353  (81)      7365  (81)
                                                    7400  (81)      7437  (81)      7562  (82)      7600  (82)      7716  (83)
                                                    7728  (83)      7851  (84)      7888  (84)      8005  (85)      8017  (85)
                                                    8137  (86)      8173  (86)      8283  (87)      8294  (87)      8477  (88)
                                                    8515  (88)      8562  (88)      8574  (88)      8755  (89)      8792  (89)
```

```
                                                              8837     (89)        8848     (89)        9023     (90)        9065     (90)        9109     (90)
                                                              9125     (90)        9297     (91)        9337     (91)        9379     (91)        9394     (91)
                                                              9520     (92)        9539     (92)        9568     (92)        9671     (93)        9686     (93)
                                                              9696     (93)        9786     (94)        9804     (94)        9822     (94)
$DS_CLRVEC_S           1                4020     (59)         4020     (59)        6675     (74)        6729     (74)        6730     (74)        6731     (74)
                                                              7109     (79)        9763     (94)        9818     (94)        9819     (94)        9820     (94)
$DS_DSDEF              2                90       (2)          90       (2)
$DS_DSSDEF             8                91       (2)          91       (2)
$DS_ENDDATA            1                3668     (56)         3668     (56)        4059     (60)        6757     (76)        7148     (80)
$DS_ENDMESSAGE         1                3174     (53)         3174     (53)        3179     (53)        3184     (53)        3201     (53)        3265     (53)
                                                              3272     (53)        3279     (53)        3311     (53)        3317     (53)        3322     (53)
$DS_ENDMOD             1                9823     (94)         9823     (94)
$DS_ENDSUB             1                3815     (57)         3815     (57)        3917     (58)        4019     (59)        4358     (61)        4519     (62)
                                                              4648     (63)        4807     (64)        4935     (65)        5092     (66)        5213     (67)
                                                              5497     (68)        5771     (69)        6046     (70)        6315     (71)        6487     (72)
                                                              6616     (73)        6727     (74)        6904     (77)        7006     (78)        7108     (79)
                                                              7447     (81)        7608     (82)        7737     (83)        7896     (84)        8024     (85)
                                                              8181     (86)        8302     (87)        8586     (88)        8860     (89)        9135     (90)
                                                              9404     (91)        9576     (92)        9704     (93)        9816     (94)
$DS_ENDTEST            1                4022     (59)         4022     (59)        6733     (74)        7111     (79)        9822     (94)
$DS_ENVDEF             2                72       (2)          72       (2)
$DS_ERRDEF             1                89       (2)          89       (2)
$DS_ERRHARD_S          1                3687     (56)         3687     (56)        3694     (56)        3708     (56)        3786     (57)        3800     (57)
                                                              3810     (57)        3888     (58)        3902     (58)        3912     (58)        3990     (59)
                                                              4004     (59)        4014     (59)        4078     (60)        4085     (60)        4099     (60)
                                                              4243     (61)        4254     (61)        4267     (61)        4288     (61)        4303     (61)
                                                              4314     (61)        4324     (61)        4340     (61)        4351     (61)        4451     (62)
                                                              4465     (62)        4476     (62)        4486     (62)        4503     (62)        4514     (62)
                                                              4604     (63)        4618     (63)        4630     (63)        4642     (63)        4740     (64)
                                                              4754     (64)        4765     (64)        4775     (64)        4791     (64)        4802     (64)
                                                              4893     (65)        4907     (65)        4919     (65)        4931     (65)        5027     (66)
                                                              5040     (66)        5051     (66)        5061     (66)        5076     (66)        5087     (66)
                                                              5172     (67)        5186     (67)        5197     (67)        5208     (67)        5366     (68)
                                                              5380     (68)        5391     (68)        5401     (68)        5418     (68)        5429     (68)
                                                              5450     (68)        5464     (68)        5476     (68)        5488     (68)        5645     (69)
                                                              5658     (69)        5669     (69)        5679     (69)        5695     (69)        5706     (69)
                                                              5726     (69)        5740     (69)        5751     (69)        5762     (69)        5912     (70)
                                                              5926     (70)        5937     (70)        5947     (70)        5968     (70)        5979     (70)
                                                              5994     (70)        6011     (70)        6023     (70)        6039     (70)        6187     (71)
                                                              6200     (71)        6211     (71)        6221     (71)        6240     (71)        6251     (71)
                                                              6265     (71)        6282     (71)        6293     (71)        6308     (71)        6410     (72)
                                                              6423     (72)        6434     (72)        6456     (72)        6471     (72)        6482     (72)
                                                              6574     (73)        6592     (73)        6601     (73)        6611     (73)        6685     (74)
                                                              6702     (74)        6719     (74)        6776     (76)        6783     (76)        6797     (76)
                                                              6875     (77)        6889     (77)        6899     (77)        6977     (78)        6991     (78)
                                                              7001     (78)        7079     (79)        7093     (79)        7103     (79)        7167     (80)
                                                              7174     (80)        7188     (80)        7332     (81)        7343     (81)        7356     (81)
                                                              7377     (81)        7392     (81)        7403     (81)        7413     (81)        7429     (81)
                                                              7440     (81)        7540     (82)        7554     (82)        7565     (82)        7575     (82)
                                                              7592     (82)        7603     (82)        7693     (83)        7707     (83)        7719     (83)
                                                              7731     (83)        7829     (84)        7843     (84)        7854     (84)        7864     (84)
                                                              7880     (84)        7891     (84)        7982     (85)        7996     (85)        8008     (85)
                                                              8020     (85)        8116     (86)        8129     (86)        8140     (86)        8150     (86)
                                                              8165     (86)        8176     (86)        8261     (87)        8275     (87)        8286     (87)
                                                              8297     (87)        8455     (88)        8469     (88)        8480     (88)        8490     (88)
```

```
                                                8507   (88)      8518   (88)      8539   (88)      8553   (88)      8565   (88)
                                                8577   (88)      8734   (89)      8747   (89)      8758   (89)      8768   (89)
                                                8784   (89)      8795   (89)      8815   (89)      8829   (89)      8840   (89)
                                                8851   (89)      9001   (90)      9015   (90)      9026   (90)      9036   (90)
                                                9057   (90)      9068   (90)      9083   (90)      9100   (90)      9112   (90)
                                                9128   (90)      9276   (91)      9289   (91)      9300   (91)      9310   (91)
                                                9329   (91)      9340   (91)      9354   (91)      9371   (91)      9382   (91)
                                                9397   (91)      9499   (92)      9512   (92)      9523   (92)      9545   (92)
                                                9560   (92)      9571   (92)      9662   (93)      9680   (93)      9689   (93)
                                                9699   (93)      9773   (94)      9790   (94)      9808   (94)
$DS_EXIT          1            4279   (61)      4279   (61)      4281   (61)      7368   (81)      7370   (81)
$DS_PAGE          1            3646   (54)      3646   (54)      3713   (56)      3817   (57)      3919   (58)      4024   (59)
                                                4104   (60)      4360   (61)      4521   (62)      4650   (63)      4809   (64)
                                                4936   (65)      5094   (66)      5215   (67)      5499   (68)      5773   (69)
                                                6048   (70)      6316   (71)      6488   (72)      6617   (73)      6735   (74)
                                                6802   (76)      6906   (77)      7008   (78)      7113   (79)      7193   (80)
                                                7449   (81)      7610   (82)      7739   (83)      7898   (84)      8025   (85)
                                                8183   (86)      8304   (87)      8588   (88)      8862   (89)      9137   (90)
                                                9405   (91)      9577   (92)      9705   (93)
$DS_PARDEF        1            92     (2)       92     (2)
$DS_PRINTF_S      1            3173   (53)      3173   (53)      3178   (53)      3183   (53)
$DS_PRINTX_S      1            3200   (53)      3200   (53)      3262   (53)      3263   (53)      3264   (53)      3269   (53)
                                                3270   (53)      3271   (53)      3276   (53)      3290   (53)      3292   (53)
                                                3294   (53)      3300   (53)      3306   (53)      3309   (53)      3310   (53)
                                                3316   (53)      3321   (53)      3610   (54)      3614   (54)      3616   (54)
                                                3626   (54)
$DS_SBTTL         1            3649   (56)      3649   (56)      3715   (57)      3819   (58)      3921   (59)      4026   (60)
                                                4106   (61)      4362   (62)      4523   (63)      4652   (64)      4811   (65)
                                                4938   (66)      5096   (67)      5217   (68)      5501   (69)      5775   (70)
                                                6050   (71)      6318   (72)      6490   (73)      6619   (74)      6738   (76)
                                                6804   (77)      6908   (78)      7010   (79)      7115   (80)      7195   (81)
                                                7451   (82)      7612   (83)      7741   (84)      7900   (85)      8027   (86)
                                                8185   (87)      8306   (88)      8590   (89)      8864   (90)      9139   (91)
                                                9407   (92)      9579   (93)      9707   (94)
$DS SCBDEF        3            93     (2)       93     (2)
$DS_SECDEF        1            94     (2)       94     (2)
$DS_SETIPL_S      1            6692   (74)      6692   (74)      6714   (74)      6725   (74)      9780   (94)      9803   (94)
                                                9814   (94)

$DS_SETVEC_S      1            3674   (56)      3674   (56)      4064   (60)      6397   (72)      6560   (73)      6676   (74)
                                                6763   (76)      7153   (80)      9486   (92)      9649   (93)      9764   (94)

$EQU              1            92     (2)       72     (2)       90     (2)       92     (2)
$EQULS1           1            90     (2)       72     (2)       90     (2)
$EQULST           1            72     (2)       72     (2)       90     (2)
$GBLINI           2            72     (2)       72     (2)       90     (2)       92     (2)
$OFFDEF           1            89     (2)       89     (2)
$PR750DEF         1            88     (2)       88     (2)
$PUSHADR          1            3687   (56)      3687   (56)      3695   (56)      3708   (56)      3786   (57)      3801   (57)
                                                3810   (57)      3888   (58)      3903   (58)      3912   (58)      3990   (59)
                                                4005   (59)      4014   (59)      4078   (60)      4086   (60)      4099   (60)
                                                4243   (61)      4254   (61)      4267   (61)      4280   (61)      4288   (61)
                                                4304   (61)      4314   (61)      4324   (61)      4341   (61)      4351   (61)
                                                4451   (62)      4466   (62)      4476   (62)      4486   (62)      4504   (62)
                                                4514   (62)      4604   (63)      4619   (63)      4630   (63)      4642   (63)
                                                4740   (64)      4755   (64)      4765   (64)      4775   (64)      4792   (64)
                                                4802   (64)      4893   (65)      4908   (65)      4919   (65)      4931   (65)
```

```
                                                 5027   (66)      5041   (66)      5051   (66)      5061   (66)      5077   (66)
                                                 5087   (66)      5172   (67)      5187   (67)      5197   (67)      5208   (67)
                                                 5366   (68)      5381   (68)      5391   (68)      5401   (68)      5419   (68)
                                                 5429   (68)      5450   (68)      5465   (68)      5476   (68)      5488   (68)
                                                 5645   (69)      5659   (69)      5669   (69)      5679   (69)      5696   (69)
                                                 5706   (69)      5726   (69)      5741   (69)      5751   (69)      5762   (69)
                                                 5912   (70)      5927   (70)      5937   (70)      5947   (70)      5969   (70)
                                                 5979   (70)      5994   (70)      6012   (70)      6023   (70)      6039   (70)
                                                 6187   (71)      6201   (71)      6211   (71)      6221   (71)      6241   (71)
                                                 6251   (71)      6265   (71)      6283   (71)      6293   (71)      6308   (71)
                                                 6410   (72)      6424   (72)      6434   (72)      6456   (72)      6472   (72)
                                                 6482   (72)      6574   (73)      6593   (73)      6601   (73)      6611   (73)
                                                 6685   (74)      6702   (74)      6719   (74)      6776   (76)      6784   (76)
                                                 6797   (76)      6875   (77)      6890   (77)      6899   (77)      6977   (78)
                                                 6992   (78)      7001   (78)      7079   (79)      7094   (79)      7103   (79)
                                                 7167   (80)      7175   (80)      7188   (80)      7332   (81)      7343   (81)
                                                 7356   (81)      7369   (81)      7377   (81)      7393   (81)      7403   (81)
                                                 7413   (81)      7430   (81)      7440   (81)      7540   (82)      7555   (82)
                                                 7565   (82)      7575   (82)      7593   (82)      7603   (82)      7693   (83)
                                                 7708   (83)      7719   (83)      7731   (83)      7829   (84)      7844   (84)
                                                 7854   (84)      7864   (84)      7881   (84)      7891   (84)      7982   (85)
                                                 7997   (85)      8008   (85)      8020   (85)      8116   (86)      8130   (86)
                                                 8140   (86)      8150   (86)      8166   (86)      8176   (86)      8261   (87)
                                                 8276   (87)      8286   (87)      8297   (87)      8455   (88)      8470   (88)
                                                 8480   (88)      8490   (88)      8508   (88)      8518   (88)      8539   (88)
                                                 8554   (88)      8565   (88)      8577   (88)      8734   (89)      8748   (89)
                                                 8758   (89)      8768   (89)      8785   (89)      8795   (89)      8815   (89)
                                                 8830   (89)      8840   (89)      8851   (89)      9001   (90)      9016   (90)
                                                 9026   (90)      9036   (90)      9058   (90)      9068   (90)      9083   (90)
                                                 9101   (90)      9112   (90)      9128   (90)      9276   (91)      9290   (91)
                                                 9300   (91)      9310   (91)      9330   (91)      9340   (91)      9354   (91)
                                                 9372   (91)      9382   (91)      9397   (91)      9499   (92)      9513   (92)
                                                 9523   (92)      9545   (92)      9561   (92)      9571   (92)      9662   (93)
                                                 9681   (93)      9689   (93)      9699   (93)      9773   (94)      9790   (94)
                                                 9808   (94)
$VIELD            1             72     (2)       72     (2)       92     (2)
$VIELD1           1             92     (2)       72     (2)       92     (2)
ECKAX_SECDEF      1             94     (2)       94     (2)

                                                +------------------------+
                                                ! Performance indicators !
                                                +------------------------+

Phase                    Page faults    CPU Time       Elapsed Time
- ---                    -- --- ----    --------       ------------
Initialization                    31    00:00:00.09    00:00:00.70
Command processing               898    00:00:00.67    00:00:01.90
Pass 1                          4135    00:01:15.82    00:02:24.28
Symbol table sort                  3    00:00:01.31    00:00:01.91
Pass 2                          1896    00:00:21.81    00:00:53.67
Symbol table output                3    00:00:00.40    00:00:01.29
Psect synopsis output              3    00:00:00.03    00:00:00.03
Cross-reference output           335    00:00:13.19    00:00:44.46
Assembler run totals            7307    00:01:53.33    00:04:08.25
```

```
The working set limit was 4096 pages.
1097601 bytes (2144 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 788 non-local and 593 local symbols.
9825 source lines were read in Pass 1, producing 98 object records in Pass 2.
97 pages of virtual memory were used to define 52 macros.

                                                +--------------------------+
                                                ! Macro library statistics !
                                                +--------------------------+

Macro library name                               Macros defined
------------------                               --------------
DISK$DIAGPACK02:[TAYLOR.ECKAX]ECKAX.MLB;1                  1
SYS$COMMON:[SYSLIB]DIAG.MLB;955                           45
SYS$COMMON:[SYSLIB]STARLET.MLB;2                           8
TOTALS (all libraries)                                    54

724 GETS were required to define 54 macros.

There were no errors, warnings or information messages.

MACRO/LIS=ECKAX7.LIS/CROSS ECKAX7
```

Table of contents

```
   (1)        1       ECKAX8: PATCH CONTROL STORE
   (2)       61       DECLARATIONS
   (3)      263       SUBROUTINES
   (5)      498       ERROR MESSAGES
   (7)      569       ERROR REPORTING ROUTINES
   (8)      645       TEST 13:  PCS Memory Test
   (9)      692       SUBTEST 13.1:  Data Path
   (10)     775       SUBTEST 13.2:  Row Select
   (11)     848       SUBTEST 13.3:  Address Path
   (12)     919       SUBTEST 13.4:  Moving Inversions
   (13)    1052       TEST 14:  PCS Execution Test
   (14)    1096       SUBTEST 14.1:  PCS Entry/Exit
   (15)    1158       SUBTEST 14.2:  PCS  Next Field' Addressing
   (16)    1218       SUBTEST 14.3:  PCS Parity Error ABORT
```

```
0000            .SBTTL  ECKAX8: PATCH CONTROL STORE
0000            .TITLE  ECKAX8  VAX-11/750 Specific CPU Cluster Exerciser
0000            .IDENT  /5.2/
0000
0000    ;
0000    ; Copyright (C) 1980, 1983,1986
0000    ; Digital Equipment Corporation, Maynard, Massachusetts 01754
0000    ;
0000    ; This software is furnished  under a license for use only on a single
0000    ; computer  system and  may be  copied only with  the inclusion of the
0000    ; above copyright notice.  This software, or any other copies thereof,
0000    ; may not be provided or  otherwise made available to any other person
0000    ; except for use on such system and to one who agrees to these license
0000    ; terms.  Title to and  ownership of the  software  shall at all times
0000    ; remain in DEC.
0000    ;
0000    ; The information in this software is subject to change without notice
0000    ; and should  not be construed  as a commitment  by Digital  Equipment
0000    ; Corporation.
0000    ;
0000    ; DEC assumes  no  responsibility  for  the use or  reliability of its
0000    ; software on equipment which is not supplied by DEC.
0000    ;
0000
0000    ;++
0000    ; Facility:     VAX Architecture Diagnostic.
0000    ;
0000    ; Abstract:     This module tests the patch contole store for parity
0000    ;               errors, addressing errors and sequence timing errors.
0000    ;
0000    ;
0000    ; Environment:  VAX Diagnostic Supervisor.
0000    ;
0000    ; Author:       Dave Shull      17-Sep-1982     Version 4.0
0000    ;
0000    ;
0000    ;       Dan Milleville  19-Mar-82               Version 3.4
0000    ;       Changed Test 3 exit code to accomodate temporarily a hardware problem.
0000    ;
0000    ;       Dave Shull      30-Nov-82               Version 4.0
0000    ;       Added tests 13 and 14 for coverage of PCS module.
0000    ;
0000    ;       Tai-Chun Pan    04-Mar-83               Version 5.0
0000    ;       Added test 5 for coverage of Processor Halts Architecture Undefined.
0000    ;       Added tests 11 and 12 for coverage of new ROM with MRSP of TU58.
0000    ;
0000    ;       Dan Milleville  21-Nov-83               Version 5.1
0000    ;       Fixed error in timer routine (didn't turn off timer).
0000    ;
0000    ;       Wayne R. Taylor February 1986           Version 5.2
0000    ;       Redefined IPR's to jive with starlet.
0000    ;       Modified the code for problem report Eckax5 where:
0000    ;               Tests 6 & 13 tests with cache off. They want cache on.
0000    ;               Subtest 1 does not check bit 0 of the memory under test.
0000    ;               Subtest 4 tests row 0, 1, 2, but not row 3.
```

```
0000     56 ;       Modifications were supplied by the submitter.
0000     57 ;       Search on 5.2
0000     58 ;
0000     59 ;--
```

```
                      0000     61          .SBTTL  DECLARATIONS
                      0000     62          .LIST   MEB
                      0000     63          .NLIST  CND
                  00000000     64          .PSECT  ECKAXH, PAGE, NOWRT
                      0000     65          .LIBRARY         'SYS$LIBRARY:DIAG ; VAX family diagnostic library.
                      0000     66          $DS_BGNMOD       CEP_REPAIR, TN=13
                      0000        ;;;      Macro-32 Diagnostic macro library Version 8.3 'DIAG.MLB (245)'
                      0000     67
                      0000     68 ;
                      0000     69 ; Include files:
                      0000     70 ;
                      0000     71
                      0000     72          .LIBRARY         'ECKAX'          ; CPU Cluster Exerciser library.
                      0000     73
                      0000     77
                      0000     78          $DS_SCBDEF
                      0000     79          $DS_DSADEF
                      0000     80          $DS_DSDEF
                      0000     81          $DS_PARDEF
                      0000     82          $DS_PSLDEF
                      0000     83          $DS_ERRDEF
                      0000     84          $DS_DSSDEF
                      0000     85          $DS_HPODEF
                      0000     86          $DS_KA_DEF
                      0000     87          ECKAX_SECDEF
                      0000     88          $pr750def        ; 5.2
                      0000     89 ;
                      0000     90 ; Own storage:
                      0000     91 ;
                      0000     92
            00000004  0000     93 PCS$L_COUNT:     .BLKL                     ; used as a counter
            00000008  0004     94 PCS$L_INC:       .BLKL                     ; used to store increment
            0000000C  0008     95 PCS$L_ROW:       .BLKL                     ; used to store current row
            00000014  000C     96 PCS$A_RETURN:    .BLKL   2                 ; used to store return address
                      0014     97
            00000018  0014     98 PCS$L_FLAGS:     .BLKL                     ; contains the following flags
                      0018     99
            00000000  0018    100 PCS$V_PAT=0                                ; pattern done flag
            00000001  0018    101 PCS$M_PAT=1@0                              ;
            00000001  0018    102 PCS$V_DATA_ERR=1                           ; data error flag
            00000002  0018    103 PCS$M_DATA_ERR=1@1                         ;
            00000002  0018    104 PCS$V_ROW_ERR=2                            ; row error flag
            00000004  0018    105 PCS$M_ROW_ERR=1@2                          ;
            00000003  0018    106 PCS$V_REVERSE=3                            ; reverse direction flag
            00000008  0018    107 PCS$M_REVERSE=1@3                          ;
            00000005  0018    108 PCS$V_EXP_CS_PAR=5                         ; Expect Control Store Parity Error
            00000020  0018    109 PCS$M_EXP_CS_PAR=1@5                       ;
                      0018    110
                      0018    111
            00000060  0018    112 SCB$L_WTBUSERR=^X60                        ; write bus interrupt vector offset
            0000000E  0018    113 PCS$V_BIT14=14                             ; bit position for bit 14
            00080000  0018    114 PCS$M_BIT19=1@19                           ; bit mask for bit 19
            00100000  0018    115 PCS$M_BIT20=1@20                           ; bit mask for bit 20
                      0018    116
                      0018    117 PCS$A_PCS_ADDRESS:
```

```
00F08000  0018    118                   .LONG    ^XF08000         ; used to store address of PCS memory
          001C    119
          001C    120 PCS$A_ENA_ADDRESS:
00F0C000  001C    121                   .LONG    ^XF0C000         ; address of PCS/Patch enable register
          0020    122
          0020    123 PCS$A_PTH_LOCATION:
00F00C00  0020    124                   .LONG    ^XF00C00         ; location to set patch bit at
          0024    125
          0024    126 PCS$A_PTHBIT_ADDRESS:
00F00000  0024    127                   .LONG    ^XF00000         ; address of start of patch bits
          0028    128
          0028    129 PCS$DATA_PATTERN:
00200000  0028    130                   .LONG    ^X200000         ; floating one pattern
FFDFFFFF  002C    131                   .LONG    ^XFFDFFFFF       ; floating zero pattern
          0030    132
000000B0  0030    133 PCS$A_PCS_BUF:  .BLKL    32               ; area used to store data
          00B0    134
          00B0    135 PCS$DECODED_WORDS:
00000130  00B0    136                   .BLKL    32               ; area used to store decoded data words
          0130    137
          0130    138 PCS$NOISY_WORDS:
000001B0  0130    139                   .BLKL    32               ; area used to store encoded data words
          01B0    140
          01B0    141 PCS$CODEVECTORS:
0000FFFF  01B0    142                   .LONG    ^XFFFF           ; identity vector
00005555  01B4    143                   .LONG    ^X5555           ; vector 1
00003333  01B8    144                   .LONG    ^X3333           ; vector 2
00000F0F  01BC    145                   .LONG    ^X0F0F           ; vector 3
          01C0    146
00000000  01C0    147 PCS$CODE_WORDS: .LONG    ^X0000           ; encoded 0
0000FFFF  01C4    148                   .LONG    ^XFFFF           ; encoded 1
00005555  01C8    149                   .LONG    ^X5555           ; encoded 2
0000AAAA  01CC    150                   .LONG    ^XAAAA           ; encoded 3
00003333  01D0    151                   .LONG    ^X3333           ; encoded 4
0000CCCC  01D4    152                   .LONG    ^XCCCC           ; encoded 5
00006666  01D8    153                   .LONG    ^X6666           ; encoded 6
00009999  01DC    154                   .LONG    ^X9999           ; encoded 7
00000F0F  01E0    155                   .LONG    ^X0F0F           ; encoded 8
0000F0F0  01E4    156                   .LONG    ^XF0F0           ; encoded 9
00005A5A  01E8    157                   .LONG    ^X5A5A           ; encoded 10(decimal)
          01EC    158
00000000  01EC    159 PCS$DATA_TABLE: .LONG    ^X00000
00011111  01F0    160                   .LONG    ^X11111
00033333  01F4    161                   .LONG    ^X33333
00077777  01F8    162                   .LONG    ^X77777
000FFFFF  01FC    163                   .LONG    ^XFFFFF
000EEEEE  0200    164                   .LONG    ^XEEEEE
000CCCCC  0204    165                   .LONG    ^XCCCCC
00088888  0208    166                   .LONG    ^X88888
00000000  020C    167                   .LONG    ^X00000
          0210    168
          0210    169
          0210    170 PCS$TABLE1:
0000000E  0210    171                   .LONG    14               ; Length for printsig
00660088  0214    172                   .LONG    DS$_MCHK         ; Type of exception
```

```
                          00000028  0218    173                  .LONG    ^X28              ; Length of stack logout
                          00000001  021C    174                  .LONG    ^X1               ; Error code
                          00000000  0220    175                  .LONG    ^X0               ; VA register
                          0000029C' 0224    176                  .LONG    PCS$PTH_INST+1    ; PC at time or error
                          000017C1  0228    177                  .LONG    ^X17C1            ; Memory Data Register
                          00000004  022C    178                  .LONG    ^X4               ; Saved Mode Register
                          00000000  0230    179                  .LONG    ^X0               ; read lock timeout register
                          00000000  0234    180                  .LONG    ^X0               ; TB group parity error register
                          00000000  0238    181                  .LONG    ^X0               ; CACHE error register
                          00000000  023C    182                  .LONG    ^X0               ; Bus Error Register
                          00000000  0240    183                  .LONG    ^X0               ; Machine Check error summary register
                          0000029B' 0244    184                  .LONG    PCS$PTH_INST      ; Interrupt / Exception PC
                          04000000  0248    185                  .LONG    ^X04000000        ; Interrupt / Exception PSL
                                    024C    186
                                    024C    187 ;
                                    024C    188 ; The following miro code data is used to test the functionlity of the Patch
                                    024C    189 ; Control Store.  After loading this microcode into PCS RAM the following
                                    024C    190 ; steps are performed to execute from PCS:
                                    024C    191 ;
                                    024C    192 ;         1.  The patch bit for CCS location 300 is set ( this is the location
                                    024C    193 ;             for the NOP opcode).  The next address field for location 300 is
                                    024C    194 ;             0002.
                                    024C    195 ;         2.  A NOP instruction will be executed and when the microcode finishes
                                    024C    196 ;             at location 300, instead of going to 0002 (because the patch bit
                                    024C    197 ;             is set), control will go to location 2802.
                                    024C    198 ;         3.  At this point execution depends on whether PCS was loaded with
                                    024C    199 ;             just the first 3 micro words or more.  The specific test will
                                    024C    200 ;             determine what the state of PCS is.
                                    024C    201 ;
                                    024C    202 PCS$MICRO_DATA:
                                    024C    203                                             ; entry and exit micro words
                                    024C    204                                             ; next field inverted in the first 20
                                    024C    205                                             ; bits.
                          000097C1  024C    206                  .LONG    ^X000097C1        ; First microword:
                          00054047  0250    207                  .LONG    ^X00054047        ;       R[R5]_D,DTYPE/LONG,NEXT/2801
                          000B2402  0254    208                  .LONG    ^X000B2402
                          00008405  0258    209                  .LONG    ^X00008405
                                    025C    210
                          00003FC2  025C    211                  .LONG    ^X00003FC2        ; Second Microword:
                          00000047  0260    212                  .LONG    ^X00000047        ;       NOP,NEXT/0002
                          00036403  0264    213                  .LONG    ^X00036403        ; Exit Microword
                          000C8000  0268    214                  .LONG    ^X000C8000
                                    026C    215
                          000097C3  026C    216                  .LONG    ^X000097C3        ; Third Microword:
                          00000047  0270    217                  .LONG    ^X00000047        ;       NOP,CLKX/XTND,NEXT/2803
                          00036403  0274    218                  .LONG    ^X00036403
                          000C8000  0278    219                  .LONG    ^X000C8000
                                    027C    220
                          000017C4  027C    221                  .LONG    ^X000017C4        ; Fourth Microword:
                          00000047  0280    222                  .LONG    ^X00000047        ;       NOP,NEXT/2804
                          00036403  0284    223                  .LONG    ^X00036403
                          000C8000  0288    224                  .LONG    ^X000C8000
                                    028C    225
                                    028C    226         ;
                                    028C    227         ; The following is the entry micro word.  When loading the micro words
```

```
                                  028C    228        ; if there is to be micro words loaded after the this location then
                                  028C    229        ; the next address (normally the first .LONG) of 2800 is changed to
                                  028C    230        ; a next address of 2803 (the fifth .LONG).
                                  028C    231        ;
                        000097C0  028C    232                .LONG   ^X000097C0      ; Fifth Microword:
                        00054047  0290    233                .LONG   ^X00054047      ;       D_R[R5]-1,DTYPE/LONG,NEXT/2800
                        0007D202  0294    234                .LONG   ^X0007D202      ; Entry Microword
                        0008800E  0298    235                .LONG   ^X0008800E
                        000097C5  029C    236                .LONG   ^X000097C5
                                  02A0    237
                                  02A0    238                                        ; these used to build rest of u words
                                  02A0    239                                        ; next field not yet inverted and last
                                  02A0    240                                        ; section clears parity depending on
                                  02A0    241                                        ; next field odd or even.
                                  02A0    242                                        ; Common Microword:
                                  02A0    243                                        ;       D_D-1,DTYPE/LONG,NEXT/28xx
                        00000047  02A0    244                .LONG   ^X00000047      ; common for second section of u word
                        00071202  02A4    245                .LONG   ^X00071202      ; common for third section of u word
                        0000A806  02A8    246                .LONG   ^X0000A806      ; next field for filling PCS
                        000C800E  02AC    247                .LONG   ^X000C800E      ; fourth section of u word
                        0000A800  02B0    248                .LONG   ^X0000A800      ; next field for going to exit point
                                  02B4    249
                                  02B4    250 PCS$PAR_ERR_WORD:
                                  02B4    251
                        00088405  02B4    252                .LONG   ^X00088405      ; This microword has the P2 bit clear
                                  02B8    253                                        ; which when used as the Fourth section
                                  02B8    254                                        ; of the first microword (reference the
                                  02B8    255                                        ; previous microwords) causes a Control
                                  02B8    256                                        ; store parity error.
                                  02B8    257
                                  02B8    258
```

```
                              02B8   263 .SBTTL  SUBROUTINES
                              02B8   264
                              02B8   265
                              02B8   266 ; This subroutine is used to clear the data buffer areas in the beginning
                              02B8   267 ; of this program.
                              02B8   268
                              02B8   269
                              02B8   270 PCS$CLEAR_BUFFER:
            55   FD74 CF  DE  02B8   271         MOVAL   PCS$A_PCS_BUF,R5          ; get starting buffer address
                 FD3F CF  D4  02BD   272         CLRL    PCS$L_COUNT               ; initialize counter
                      85  D4  02C1   273 1$:     CLRL    (R5)+                     ; clear a buffer location
F4 FD34 CF   00000060 8F  F3  02C3   274         AOBLEQ  #96,PCS$L_COUNT,1$        ; br = not finished
                          05  02CD   275         RSB                               ; return
                              02CE   276
                              02CE   277 ;
                              02CE   278 ; This routine is used to initailly zero PCS
                              02CE   279 ;
                              02CE   280
                              02CE   281 PCS$CLEAR_PCS:
            55   FD52 CF  D0  02CE   282         MOVL    PCS$A_PTHBIT_ADDRESS,R5 ; get starting address of patch bits
            FD44 DF   01  D0  02D3   283         MOVL    #1,@PCS$A_ENA_ADDRESS     ; Enable writing of patch bits
                 FD24 CF  D4  02D8   284         CLRL    PCS$L_COUNT               ; clear the counter
                 85   01  D0  02DC   285 10$:    MOVL    #1,(R5)+                  ; Disable the patch bit
F3 FD18 CF   00001FFF 8F  F3  02DF   286         AOBLEQ  #8191,PCS$L_COUNT,10$     ; br = not finished
            55   FD2B CF  D0  02E9   287         MOVL    PCS$A_PCS_ADDRESS,R5      ; get starting PCS address
                 FD2A DF  D4  02EE   288         CLRL    @PCS$A_ENA_ADDRESS        ; Enable writing of PCS
                 FD0A CF  D4  02F2   289         CLRL    PCS$L_COUNT               ; initialize counter
                      85  D4  02F6   290 20$:    CLRL    (R5)+                     ; clear a buffer location
F4 FCFF CF   00000FFF 8F  F3  02F8   291         AOBLEQ  #4095,PCS$L_COUNT,20$     ; br = not finished
                          05  0302   292         RSB                               ; return
                              0303   293
                              0303   294
                              0303   295 ; This routine is used to decode data read from the PCS that has been encoded
                              0303   296 ; in the REED - MULLER code for the address path subtest.
                              0303   297
                              0303   298 PCS$RM_DECODE:
                 03FF 8F  BB  0303   299         PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9>
                      52  D4  0307   300         CLRL    R2                        ; initialize word counter
                 54   0B  D0  0309   301         MOVL    #11,R4                    ; initialize number of words to decode
               FD9F CF42  D4  030C   302 10$:    CLRL    PCS$DECODED_WORDS[R2]     ; initialize decoded word storage
                 53   01  D0  0311   303         MOVL    #1,R3                     ; initialize the pair seperation
                 55   01  D0  0314   304         MOVL    #1,R5                     ; initialize the message bit counter
                 56   0F  D0  0317   305 20$:    MOVL    #15,R6                    ; initialize the code word bit position
                      58  D4  031A   306         CLRL    R8                        ; initialize the population counter
                 55   01  D1  031C   307         CMPL    #1,R5                     ; if decoding bit one, skip next line
                      04  13  031F   308         BEQL    30$                       ; br = do not update pair seperation
            53   53   01  78  0321   309         ASHL    #1,R3,R3                  ; update pair seperation
          57   FE06 CF42  D0  0325   310 30$:    MOVL    PCS$NOISY_WORDS[R2],R7    ; get the encoded word
              05 57   56  E1  032B   311         BBC     R6,R7,40$                 ; br = codeword bit not set and store zero
                 50   01  D0  032F   312         MOVL    #1,R0                     ; store a one for bit set in first of pair
                      02  11  0332   313         BRB     50$                       ; skip
                      50  D4  0334   314 40$:    CLRL    R0                        ; store a zero for bit clear
                 56   53  C2  0336   315 50$:    SUBL2   R3,R6                     ; point to the next bit of the pair
              05 57   56  E1  0339   316         BBC     R6,R7,60$                 ; br = codeword bit not set and store zero
                 51   01  D0  033D   317         MOVL    #1,R1                     ; store a one for bit set in second of pair
```

```
                02  11  0340          BRB     70$                     ; skip
                51  D4  0342 60$:     CLRL    R1                      ; store a zero for bit clear
           51   50  CC  0344 70$:     XORL2   R0,R1                   ; compute the value of the relation
                02  13  0347          BEQL    80$                     ; br = value is not a one
                58  D6  0349          INCL    R8                      ; bump the population counter
      50   01   55  78  034B 80$:     ASHL    R5,#1,R0                ; calculate 2**R5 for message bit
 59   00   56   01  7A  034F          EMUL    #1,R6,#0,R9             ;
 50   50   59   50  7B  0354          EDIV    R0,R9,R0,R0             ;
                50  D5  0359          TSTL    R0                      ;
                03  13  035B          BEQL    90$                     ; br =
           56   53  C0  035D          ADDL2   R3,R6                   ;
             C2 56  F4  0360 90$:     SOBGEQ  R6,30$                  ;
           04   58  D1  0363          CMPL    R8,#4                   ;
                09  15  0366          BLEQ    100$                    ;
         FD43 CF42  DF  0368          PUSHAL  PCS$DECODED_WORDS[R2]   ;
        00 9E   55  E2  036D          BBSS    R5,@(SP)+,100$          ;
        A2 55   03  F3  0371 100$:    AOBLEQ  #3,R5,20$               ; br = not finished with message bits
        93 52   54  F2  0375          AOBLSS  R4,R2,10$               ; br = not finished with all codewords
                52  D4  0379          CLRL    R2                      ; intialize word counter to do bit 0
           54   0B  D0  037B          MOVL    #11,R4                  ; initialize number of words to decode
    50   FDAD CF42  D0  037E 110$:    MOVL    PCS$NOISY_WORDS[R2],R0  ; get encoded word
           53   01  D0  0384          MOVL    #1,R3                   ; initialize bit counter
    56   FD24 CF42  D0  0387 120$:    MOVL    PCS$DECODED_WORDS[R2],R6; get a partially decoded word
 56   56   01   53  EF  038D          EXTZV   R3,#1,R6,R6             ; get a decoded bit
    56   FE19 CF43  C4  0392          MULL2   PCS$CODEVECTORS[R3],R6  ; get product of vector and decoded bit
           50   56  CC  0398          XORL2   R6,R0                   ; get XOR of each product
        E8 53   03  F3  039B          AOBLEQ  #3,R3,120$              ; do for all three decoded bits
                55  D4  039F          CLRL    R5                      ; initialize the population counter
                53  D4  03A1          CLRL    R3                      ; initialize the bit counter
        02 50   53  E1  03A3 130$:    BBC     R3,R0,140$              ; br = bit is a zero
                55  D6  03A7          INCL    R5                      ; else bump the population counter
        F6 53   0F  F3  03A9 140$:    AOBLEQ  #15,R3,130$             ; do for all 16 bits
           08   55  D1  03AD          CMPL    R5,#8                   ; make the majority decision
                09  15  03B0          BLEQ    150$                    ; br = majority not ones
         FCF9 CF42  DF  03B2          PUSHAL  PCS$DECODED_WORDS[R2]   ; prepare to set LSB
        00 9E   00  E2  03B7          BBSS    #0,@(SP)+,150$          ; set LSB in decoded word
        BF 52   54  F2  03BB 150$:    AOBLSS  R4,R2,110$              ; br = not finished with all words
           03FF 8F  BA  03BF          POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9>
                    05  03C3          RSB                             ; return
                        03C4
                        03C4
                        03C4 ; This routine is used to load a 1K buffer area with test addresses for the
                        03C4 ; moving inversions subtest. The different sequence of addresses accessed
                        03C4 ; for the current increment are computed and written into the buffer area.
                        03C4
                        03C4 PCS$SETUP_ADDR:
      57   FC3C CF  D0  03C4          MOVL    PCS$L_INC,R7            ; save increment
  58   00000000'EF  D0  03C9          MOVL    PCS$A_PCS_BUFFER,R8     ; save address of buffer to store
                        03D0                                          ; addresses
      5B   FC44 CF  D0  03D0          MOVL    PCS$A_PCS_ADDRESS,R11   ; save first address of pcs
 59   5B   FC2F CF  C1  03D5 5$:      ADDL3   PCS$L_ROW,R11,R9        ; add row being tested to address to R9
           88   59  D0  03DB 10$:     MOVL    R9,(R8)+                ; load address to use into buffer
           59   57  C0  03DE          ADDL2   R7,R9                   ; add increment to get next address
        F6 59   0E  E1  03E1          BBC     #PCS$V_BIT14,R9,10$     ; br = no address overflow yet
           59   57  C2  03E5          SUBL2   R7,R9                   ; get last address done
```

```
            5A   59   0A   04   EE  03E8   373           EXTV    #4,#10,R9,R10           ; get address by itself
            5A    FFFFFFFF 8F   D1  03ED   374           CMPL    #-1,R10                 ; check for minus one
                           05   13  03F4   375           BEQL    20$                     ; br = finished building addresses
                      5B   10   C0  03F6   376           ADDL2   #16,R11                 ; increment the base address
                           DA   11  03F9   377           BRB     5$                      ; go finish building addresses
                                05  03FB   378 20$:      RSB                             ; return
                                    03FC   379
                                    03FC   380 ; This subroutine will load PCS with executable micro code.  Three microwords
                                    03FC   381 ; are first loaded into PCS then the value of PCS$L_COUNT is tested to see if
                                    03FC   382 ; it is zero or not.  If PCS$L_COUNT is equal to zero then the test is exited
                                    03FC   383 ; else the number of microwords specified by PCS$L_COUNT is loaded (these
                                    03FC   384 ; microwords are a copy of each other and perform D_D-1,NEXT/<next_pcs_address>
                                    03FC   385 ; where the next_pcs_address is just an incremented address).  The last micro
                                    03FC   386 ; word to be loaded will have a next field of 2800.  Before exiting this
                                    03FC   387 ; routine there will be a test made on the flag bit PCS$V_EXP_CS_PAR.  If this
                                    03FC   388 ; bit is equal to 1 then the first microword in PCS is loaded with bad parity
                                    03FC   389 ; else no bad parity is loaded.
                                    03FC   390 ; For the entry/exit and PCS parity tests a count of zero is used.
                                    03FC   391
                                    03FC   392 PCS$LOAD_MICRO:
                      FC1C DF   D4  03FC   393           CLRL    @PCS$A_ENA_ADDRESS      ; enable loading of PCS
                           5A   D4  0400   394           CLRL    R10                     ; initialize a counter for first two words
                 53   FC12 CF   D0  0402   395           MOVL    PCS$A_PCS_ADDRESS,R3    ; load PCS starting address
                 54   FE41 CF   DE  0407   396           MOVAL   PCS$MICRO_DATA,R4       ; load address of micro code to be loaded
                      63   84   D0  040C   397 1$:       MOVL    (R4)+,(R3)              ; load 20 bits of micro word
                      53   04   C0  040F   398           ADDL2   #4,R3                   ; get next address
                   F6 5A   10   F2  0412   399           AOBLSS  #16,R10,1$              ; br = until first and last words are done
                      56   84   7D  0416   400           MOVQ    (R4)+,R6                ; load first two words in R6 & R7
                      58   84   7D  0419   401           MOVQ    (R4)+,R8                ; load second two words in R8 & R9
                      FBE0 CF   D5  041C   402           TSTL    PCS$L_COUNT             ; check for extra words
                           0F   12  0420   403           BNEQ    3$                      ; br = more than one micro word to load
                      83   56   D0  0422   404           MOVL    R6,(R3)+                ; write first 20 bits
                      83   57   D0  0425   405           MOVL    R7,(R3)+                ; write second 20 bits
                      83   58   D0  0428   406           MOVL    R8,(R3)+                ; write third 20 bits
                      83   59   D0  042B   407           MOVL    R9,(R3)+                ; write fourth 20 bits
                         0060   31  042E   408           BRW     40$                     ; br = finished
                      83   84   D0  0431   409 3$:       MOVL    (R4)+,(R3)+             ; write the first 20 bits
                      83   57   D0  0434   410           MOVL    R7,(R3)+                ; write second 20 bits
                      83   58   D0  0437   411           MOVL    R8,(R3)+                ; write third 20 bits
                      83   59   D0  043A   412           MOVL    R9,(R3)+                ; write fourth 20 bits
                           56   D4  043D   413           CLRL    R6                      ; initialize next address field for
                                    043F   414                                           ; other words
                      57   84   7D  043F   415           MOVQ    (R4)+,R7                ; load constant second 40 bits
                      56   84   D0  0442   416           MOVL    (R4)+,R6                ; setup next address field
                           50   7C  0445   417 5$:       CLRQ    R0                      ; init bit and count to figure high
                                    0447   418                                           ; parity bit
                      59   64   D0  0447   419           MOVL    (R4),R9                 ; load last 20 bits, parity bit is
                                    044A   420                                           ; clear
                   02 56   50   E1  044A   421 7$:       BBC     R0,R6,10$               ; br = bit is zero do not count
                           51   D6  044E   422           INCL    R1                      ; bit non zero, count
                   F6 50   0C   F2  0450   423 10$:      AOBLSS  #12,R0,7$               ; br = not finished checking next
                                    0454   424                                           ; address field
                   07 51   00   E0  0454   425           BBS     #0,R1,20$               ; br = Odd number of bits,
                                    0458   426                                           ; do not clear parity bit
            59    00080000 8F   CA  0458   427           BICL2   #PCS$M_BIT19,R9         ; clear high parity bit
```

```
                  52    56    D0  045F    428 20$:    MOVL    R6,R2                     ; load next address field into R2
         5B   52  08    06    EF  0462    429         EXTZV   #6,#8,R2,R11              ; get part of next field to invert
                  5B    5B    B2  0467    430         MCOMW   R11,R11                   ; invert the data
         52   08  06    5B    F0  046A    431         INSV    R11,#6,#8,R2              ; put complete next address field
                                  046F    432                                           ; together
                  63    52    D0  046F    433         MOVL    R2,(R3)                   ; write first 20 bits
               04 A3    57    D0  0472    434         MOVL    R7,4(R3)                  ; write second 20 bits
                8 A3    58    D0  0476    435         MOVL    R8,8(R3)                  ; write third 20 bits
               0C A3    59    D0  047A    436         MOVL    R9,12(R3)                 ; write last 20 bits
                  53    10    C0  047E    437         ADDL2   #16,R3                    ; update address for next word
                        56    D6  0481    438         INCL    R6                        ; setup next address field
               BD FB79 CF     F5  0483    439         SOBGTR  PCS$L_COUNT,5$            ; br = load next micro word
                        07    19  0488    440         BLSS    40$                       ; br = finished
               56    04 A4    D0  048A    441 30$:    MOVL    4(R4),R6                  ; setup last next address field
                      FFB4    31  048E    442         BRW     5$                        ; go load it
          0A FB7E CF    05    E0  0491    443 40$:    BBS     #PCS$V_EXP_CS_PAR,PCS$L_FLAGS,50$
                                  0497    444                                           ; Check to see if we need to load a
                                  0497    445                                           ; parity error word in PCS
      FB78 DF    00100000 8F  C8  0497    446         BISL2   #PCS$M_BIT20,@PCS$A_PCS_ADDRESS
                                  04A0    447                                           ; Enable PCS
                              05  04A0    448         RSB                               ; return
            53    FB73 CF     D0  04A1    449 50$:    MOVL    PCS$A_PCS_ADDRESS,R3      ; load PCS starting address
         0C A3    FE0A CF     D0  04A6    450         MOVL    PCS$PAR_ERR_WORD,12(R3) ; load word with inverted P2 bit
      FB63 DF    00100000 8F  C8  04AC    451         BISL2   #PCS$M_BIT20,@PCS$A_PCS_ADDRESS
                                  04B5    452                                           ; Enable PCS
                              05  04B5    453         RSB                               ; return
                                  04B6    454
                                  04B6    455
                                  04B6    456 ; This routine is used to service expected and unexpected machine check
                                  04B6    457 ; exceptions. Unexpected machine check exceptions are resignaled and
                                  04B6    458 ; reported by the Supervisor.  This routine will check a flag PCS$M_EXP_CS_PAR
                                  04B6    459 ; to see if there is a CS parity error machine check expected and if yes but
                                  04B6    460 ; the machine check was of a different nature then the stack is stored in a
                                  04B6    461 ; buffer for error logout.  This is performed for the PCS Parity Error test
                                  04B6    462 ; which expected a parity error.
                                  04B6    463
                                  04B6    464
                                  04B6    465 .ALIGN  LONG
                                  04B8    466
                                  04B8    467 PCS$CHECK_MCHK:
               04 AE    01    D1  04B8    468         CMPL    #1,4(SP)                  ; check for CS parity error
                        1C    13  04BC    469         BEQL    10$                       ; br = yes parity error
          1F FB51 CF    05    E0  04BE    470         BBS     #PCS$V_EXP_CS_PAR,PCS$L_FLAGS,20$
                                  04C4    471                                           ; Br = didn't get a cs parity error but
                                  04C4    472                                           ; expected one
                        0E    DD  04C4    473         PUSHL   #14                       ; put size on stack for printsig
             00660088'EF      DD  04C6    474         PUSHL   DS$_MCHK                  ; and type
                                  04CC    475         $DS_PRINTSIG_G ARGPTR=(SP)
           00010100 9F  6E    FA  04CC                        CALLG   (SP), @#DS$PRINTSIG
                                  04D3    476         $DS_ABORT
           00010020 9F  6C    FA  04D3                        CALLG   (AP), @#DS$ABORT
                                  04DA    477 ;
                                  04DA    478 ; Come here when there was a CS Parity Error
                                  04DA    479 ;
                  5E    8E    C0  04DA    480 10$:    ADDL2   (SP)+,SP                  ; get over machine check logout
```

```
                 6E    FB2B CF   D0  04DD    481          MOVL    PCS$A_RETURN,(SP)        ; set up return address
                                 02  04E2    482          REI                              ; return
                                     04E3    483 ;
                                     04E3    484 ; Come here when we expected a CS Parity Error ( flag PCS$V_EXP_CS_PAR set) but
                                     04E3    485 ; didn't get one.
                                     04E3    486 ;
              51    00000000'EF  D0  04E3    487 20$:     MOVL    A_MCHK_BUFFER,R1         ; Load R1 with address of buffer
                       51    08  C0  04EA    488          ADDL2   #8,R1                    ; Save room for printsig size and type
                             52  D4  04ED    489          CLRL    R2                       ; Init counter
                       81    8E  D0  04EF    490 30$:     MOVL    (SP)+,(R1)+              ; Move stack data to buffer
                    F9 52    0C  F3  04F2    491          AOBLEQ  #12,R2,30$               ; Br = not finished storing stack data
                       5E    08  C2  04F6    492          SUBL2   #8,SP                    ; backup the SP for REI
                 6E    FB13 CF   D0  04F9    493          MOVL    PCS$A_RETURN+4,(SP)      ; Set-up the return address
                                 02  04FE    494          REI
                                     04FF    495
```

```
                                         04FF    497
                                         04FF    498 .SBTTL  ERROR MESSAGES
                                         04FF    499
                                         04FF    500 ;------------------------------------------------------------------
                                         04FF    501 PCS$T_CS_PARITY1:
65 20 79 74 69 72 61 70 20 53 43 00'     04FF    502         .ASCIC  CS parity error occured durring moving inversions.
64 65 72 75 63 63 6F 20 72 6F 72 72      050B
76 6F 6D 20 67 6E 69 72 72 75 64 20      0517
6F 69 73 72 65 76 6E 69 20 67 6E 69      0523
                           2E 73 6E      052F
                                 32      04FF
                                         0532    503 ;------------------------------------------------------------------
                                         0532    504 PCS$T_CS_PARITY2:
65 20 79 74 69 72 61 70 20 53 43 00'     0532    505         .ASCIC  CS parity error occured durring entry/exit.
64 65 72 75 63 63 6F 20 72 6F 72 72      053E
74 6E 65 20 67 6E 69 72 72 75 64 20      054A
            2E 74 69 78 65 2F 79 72      0556
                                 2B      0532
                                         055E    506 ;------------------------------------------------------------------
                                         055E    507 PCS$T_CS_PARITY3:
65 20 79 74 69 72 61 70 20 53 43 00'     055E    508         .ASCIC  CS parity error occured durring next address field.
64 65 72 75 63 63 6F 20 72 6F 72 72      056A
78 65 6E 20 67 6E 69 72 72 75 64 20      0576
69 66 20 73 73 65 72 64 64 61 20 74      0582
                        2E 64 6C 65      058E
                                 33      055E
                                         0592    509 ;------------------------------------------------------------------
                                         0592    510 PCS$T_MC_MESS:
20 64 65 74 70 6D 65 74 74 41 20 00'     0592    511         .ASCIC   Attempted to cause a Machine Check by!/
4D 20 61 20 65 73 75 61 63 20 6F 74      059E
6B 63 65 68 43 20 65 6E 69 68 63 61      05AA
               2F 21 79 62 20            05B6
                                 28      0592
                                         05BB    512 ;------------------------------------------------------------------
                                         05BB    513 PCS$T_PCS_ERR:
20 67 6E 69 74 75 63 65 78 65 20 00'     05BB    514         .ASCIC   executing out of PCS with bad parity!/!/
77 20 53 43 50 20 66 6F 20 74 75 6F      05C7
69 72 61 70 20 64 61 62 20 68 74 69      05D3
            2F 21 2F 21 79 74            05DF
                                 29      05BB
                                         05E5    515 ;------------------------------------------------------------------
                                         05E5    516 PCS$T_MC_ERR:
6D 20 6E 69 20 72 6F 72 72 45 20 00'     05E5    517         .ASCIC   Error in machine check logout stack!/-
6B 63 65 68 63 20 65 6E 69 68 63 61      05F1
63 61 74 73 20 74 75 6F 67 6F 6C 20      05FD
21 3A 4C 41 55 54 43 41 20 2F 21 6B      0609
                                 2F      0615
                                 30      05E5
                                         0616    518                  ACTUAL:!/
                                         0616    519 ;------------------------------------------------------------------
                                         0616    520 PCS$T_EXPECTED:
21 3A 44 45 54 43 45 50 58 45 20 00'     0616    521         .ASCIC   EXPECTED:!/
                                 2F      0622
                                 0C      0616
                                         0623    522 ;------------------------------------------------------------------
```

```
                                      0623     523 PCS$T_NO_PCS:
50 20 67 6E 69 74 69 78 45 2F 21 00'  0623     524          .ASCIC   !/Exiting PCS test, PCS is Not Present.!/'
53 43 50 20 2C 74 73 65 74 20 53 43   062F
73 65 72 50 20 74 6F 4E 20 73 69 20   063B
               2F 21 2E 74 6E 65      0647
                                29    0623
                                      064D     525 ;------------------------------------------------------------------------
                                      064D     526 PCS$T_PCS_NORESPOND:
50 20 67 6E 69 74 69 78 45 2F 21 00'  064D     527          .ASCIC   !/Exiting PCS test, PCS Address F08000(hex) didn't respond.!/'
53 43 50 20 2C 74 73 65 74 20 53 43   0659
38 30 46 20 73 73 65 72 64 64 41 20   0665
64 69 64 20 29 78 65 68 28 30 30 30   0671
2E 64 6E 6F 70 73 65 72 20 74 27 6E   067D
                             2F 21    0689
                                3D    064D
                                      068B     528 ;------------------------------------------------------------------------
                                      068B     529 PCS$T_DATAPATH:
65 72 20 61 74 61 64 20 64 61 42 00'  068B     530          .ASCIC   Bad data recieved checking data path.
6B 63 65 68 63 20 64 65 76 65 69 63   0697
74 61 70 20 61 74 61 64 20 67 6E 69   06A3
                             2E 68    06AF
                                25    068B
                                      06B1     531 ;------------------------------------------------------------------------
                                      06B1     532 PCS$T_DPERR:
21 20 3D 20 73 73 65 72 64 64 41 00'  06B1     533          .ASCIC   Address = !XL  Expected = !XL  Actual = !XL  XOR = !XL!/
64 65 74 63 65 70 78 45 20 20 4C 58   06BD
75 74 63 41 20 20 4C 58 21 20 3D 20   06C9
4F 58 20 20 4C 58 21 20 3D 20 6C 61   06D5
         2F 21 4C 58 21 20 3D 20 52   06E1
                                38    06B1
                                      06EA     534 ;------------------------------------------------------------------------
                                      06EA     535 PCS$T_ROW:
65 72 20 61 74 61 64 20 64 61 42 00'  06EA     536          .ASCIC   Bad data recieved checking row select.
6B 63 65 68 63 20 64 65 76 65 69 63   06F6
65 6C 65 73 20 77 6F 72 20 67 6E 69   0702
                          2E 74 63    070E
                                26    06EA
                                      0711     537 ;------------------------------------------------------------------------
                                      0711     538 PCS$T_ROWERR:
20 20 20 73 73 65 72 64 64 41 20 00'  0711     539          .ASCIC    Address     Expected     Actual     XOR!/
20 20 64 65 74 63 65 70 78 45 20 20   071D
20 20 20 6C 61 75 74 63 41 20 20 20   0729
            2F 21 52 4F 58 20 20      0735
                                2A    0711
                                      073C     540 ;------------------------------------------------------------------------
                                      073C     541 PCS$F_ROWERR1:
4C 58 21 20 20 20 20 4C 58 21 20 00'  073C     542          .ASCIC    !XL    !XL    !XL   !XL!/
58 21 20 20 20 4C 58 21 20 20 20 20   0748
                          2F 21 4C    0754
                                1A    073C
                                      0757     543 ;------------------------------------------------------------------------
                                      0757     544 PCS$F_ROWERR2:
4C 58 21 20 20 20 20 4C 58 21 20 00'  0757     545          .ASCIC    !XL    !XL    !XL!/
         2F 21 4C 58 21 20 20 20 20   0763
                                14    0757
```

```
                                            076C     546 ;------------------------------------------------------------------------
                                            076C     547 PCS$T_ADR_ERR:
65 64 20 61 74 61 64 20 64 61 42 00'        076C     548          .ASCIC   'Bad data decoded while checking address path.
20 65 6C 69 68 77 20 64 65 64 6F 63         0778
64 64 61 20 67 6E 69 6B 63 65 68 63         0784
      2E 68 74 61 70 20 73 73 65 72         0790
                                 2D         076C
                                            079A     549 ;------------------------------------------------------------------------
                                            079A     550 PCS$T_MI_ERR1:
6E 69 20 61 74 61 64 20 64 61 42 00'        079A     551          .ASCIC   'Bad data in PCS before writing for moving inversions.
20 65 72 6F 66 65 62 20 53 43 50 20         07A6
20 72 6F 66 20 67 6E 69 74 69 72 77         07B2
72 65 76 6E 69 20 67 6E 69 76 6F 6D         07BE
                  2E 73 6E 6F 69 73         07CA
                                 35         079A
                                            07D0     552 ;------------------------------------------------------------------------
                                            07D0     553 PCS$T_MI_ERR2:
6E 69 20 61 74 61 64 20 64 61 42 00'        07D0     554          .ASCIC   'Bad data in PCS after writing for moving inversions."
77 20 72 65 74 66 61 20 53 43 50 20         07DC
6D 20 72 6F 66 20 67 6E 69 74 69 72         07E8
73 72 65 76 6E 69 20 67 6E 69 76 6F         07F4
                     2E 73 6E 6F 69         0800
                                 34         07D0
                                            0805     555 ;------------------------------------------------------------------------
                                            0805     556 PCS$T_ENTRY_ERR:
6E 69 20 61 74 61 64 20 64 61 42 00'        0805     557          .ASCIC    Bad data in R5 executing entry/exit.
6E 69 74 75 63 65 78 65 20 35 52 20         0811
74 69 78 65 2F 79 72 74 6E 65 20 67         081D
                                 2E         0829
                                 24         0805
                                            082A     558 ;------------------------------------------------------------------------
                                            082A     559 PCS$T_NXFIELD_ERR:
6E 69 20 61 74 61 64 20 64 61 42 00'        082A     560          .ASCIC   'Bad data in R5 while checking PCS NEXT field.'
68 63 20 65 6C 69 68 77 20 35 52 20         0836
4E 20 53 43 50 20 67 6E 69 6B 63 65         0842
      2E 64 6C 65 69 66 20 54 58 45         084E
                                 2D         082A
                                            0858     561 ;------------------------------------------------------------------------
                                            0858     562 PCS$F_PCS_ERR:
45 20 2D 20 61 74 61 64 20 35 52 00'        0858     563          .ASCIC    R5 data - Expected = !XL  Actual = !XL  XOR = !XL!/"
58 21 20 3D 20 64 65 74 63 65 70 78         0864
20 3D 20 6C 61 75 74 63 41 20 20 4C         0870
21 20 3D 20 52 4F 58 20 20 4C 58 21         087C
                        2F 21 4C 58         0888
                                 33         0858
                                            088C     564 ;------------------------------------------------------------------------
                                            088C     565
                                            088C     566
```

```
                                      088C   568
                                      088C   569 .SBTTL  ERROR REPORTING ROUTINES
                                      088C   570
                                      088C   571 ;------------------------------------------------------------------------------
                                      088C   572 PCS$DPERR:
                                      088C   573        $DS_BGNMESSAGE  REGMASK=<R3,R4,R5,R6,R7>
                                00F8  088C                   .WORD   ^M<R3,R4,R5,R6,R7>              ; ENTRY MASK
          57   18 AC   14   00   EF   088E   574        EXTZV   #0,#20,ERR$_P2(AP),R7   ; get expected data
               53   1C AC   57   CD   0894   575        XORL3   R7,ERR$_P3(AP),R3       ; get XOR of data
                                      0899   576        $DS_PRINTB_S    FORMAT=PCS$T_DPERR,P1=ERR$_P1(AP),P2=R7,-
                                      0899   577                        P3=ERR$_P3(AP),P4=R3
                            53   DD   0899                   PUSHL   R3
                         1C AC   DD   089B                   PUSHL   ERR$_P3(AP)
                            57   DD   089E                   PUSHL   R7
                         14 AC   DD   08A0                   PUSHL   ERR$_P1(AP)
                       FE0A CF   9F   08A3                   PUSHAB  PCS$T_DPERR
              000100E0 9F   05   FB   08A7                   CALLS   #$$N, @#DS$PRINTB
                                      08AE   578        $DS_ENDMESSAGE
                                 04   08AE                   RET                     ; RETURN TO DIAGNOSTIC SUPERVISOR
                                      08AF   579 ;------------------------------------------------------------------------------
                                      08AF   580 PCS$ROWERR:
                                      08AF   581        $DS_BGNMESSAGE  REGMASK=<R4,R5,R8,R9>
                                0330  08AF                   .WORD   ^M<R4,R5,R8,R9>         ; ENTRY MASK
                                      08B1   582        $DS_PRINTB_S    FORMAT=PCS$T_ROWERR
                       FE5C CF   9F   08B1                   PUSHAB  PCS$T_ROWERR
              000100E0 9F   01   FB   08B5                   CALLS   #$$N, @#DS$PRINTB
                       58   01   D0   08BC   583        MOVL    #1,R8                   ; load expected data and counter
                  54   F755 CF   D0   08BF   584        MOVL    PCS$A_PCS_ADDRESS,R4    ; load initial PCS address
                  55   F768 CF   DE   08C4   585        MOVAL   PCS$A_PCS_BUF,R5        ; load address of buffer
                  59   65   58   CD   08C9   586 1$:    XORL3   R8,(R5),R9              ; generate XOR of data
                            15   13   08CD   587        BEQL    2$                      ; skip basic message if data is good
                                      08CF   588        $DS_PRINTB_S    FORMAT=PCS$F_ROWERR1,P1=R4,P2=R8,P3=(R5),P4=R9
                            59   DD   08CF                   PUSHL   R9
                            65   DD   08D1                   PUSHL   (R5)
                            58   DD   08D3                   PUSHL   R8
                            54   DD   08D5                   PUSHL   R4
                       FE61 CF   9F   08D7                   PUSHAB  PCS$F_ROWERR1
              000100E0 9F   05   FB   08DB                   CALLS   #$$N, @#DS$PRINTB
                            11   11   08E2   589        BRB     3$                      ; skip extended message
                                      08E4   590 2$:    $DS_PRINTX_S    FORMAT=PCS$F_ROWERR2,P1=R4,P2=R8,P3=(R5)
                            65   DD   08E4                   PUSHL   (R5)
                            58   DD   08E6                   PUSHL   R8
                            54   DD   08E8                   PUSHL   R4
                       FE69 CF   9F   08EA                   PUSHAB  PCS$F_ROWERR2
              000100E8 9F   04   FB   08EE                   CALLS   #$$N, @#DS$PRINTX
                            85   D5   08F5   591 3$:    TSTL    (R5)+                   ; update R5
                       54   04   C0   08F7   592        ADDL2   #4,R4                   ; update R4
                    CB 58   04   F3   08FA   593        AOBLEQ  #4,R8,1$                ; br = not finished
                                      08FE   594        $DS_ENDMESSAGE
                                 04   08FE                   RET                     ; RETURN TO DIAGNOSTIC SUPERVISOR
                                      08FF   595 ;------------------------------------------------------------------------------
                                      08FF   596 PCS$ADDR_ERR:
                                      08FF   597        $DS_BGNMESSAGE  REGMASK=<R4,R5,R6,R7,R8,R9>
                                03F0  08FF                   .WORD   ^M<R4,R5,R6,R7,R8,R9>           ; ENTRY MASK
                                      0901   598        $DS_PRINTB_S    FORMAT=PCS$T_ROWERR
          FE0C CF   9F  0901                    PUSHAB  PCS$T_ROWERR
      000100E0 9F   01  FB  0905                    CALLS   #$$N, @#DS$PRINTB
                57  D4  090C   599          CLRL    R7                          ; load expected data and counter
       54   14 AC   D0  090E   600          MOVL    ERR$_P1(AP),R4              ; load first address of row selected
            53  08  D0  0912   601          MOVL    #8,R3                       ; initialize floating address
      55  F797 CF   DE  0915   602          MOVAL   PCS$DECODED_WORDS,R5        ; load address of actual data
      56    54  53  C1  091A   603 1$:      ADDL3   R3,R4,R6                    ; build address
      59    65  57  CD  091E   604          XORL3   R7,(R5),R9                  ; generate XOR of data
                15  13  0922   605          BEQL    2$                          ; skip basic message if data is good
                        0924   606          $DS_PRINTB_S    FORMAT=PCS$F_ROWERR1,P1=R6,P2=R7,P3=(R5),P4=R9
                59  DD  0924                        PUSHL   R9
                65  DD  0926                        PUSHL   (R5)
                57  DD  0928                        PUSHL   R7
                56  DD  092A                        PUSHL   R6
          FE0C CF   9F  092C                        PUSHAB  PCS$F_ROWERR1
      000100E0 9F   05  FB  0930                    CALLS   #$$N, @#DS$PRINTB
                11  11  0937   607          BRB     3$                          ; skip extended message
                        0939   608 2$:      $DS_PRINTX_S    FORMAT=PCS$F_ROWERR2,P1=R6,P2=R7,P3=(R5)
                65  DD  0939                        PUSHL   (R5)
                57  DD  093B                        PUSHL   R7
                56  DD  093D                        PUSHL   R6
          FE14 CF   9F  093F                        PUSHAB  PCS$F_ROWERR2
      000100E8 9F   04  FB  0943                    CALLS   #$$N, @#DS$PRINTX
                85  D5  094A   609 3$:      TSTL    (R5)+                       ; update R5
      53    53  01  9C  094C   610          ROTL    #1,R3,R3                    ; float one across address
         C6 57  0A  F3  0950   611          AOBLEQ  #10,R7,1$                   ; br = not finished
                        0954   612          $DS_ENDMESSAGE
                    04  0954                        RET                         ; RETURN TO DIAGNOSTIC SUPERVISOR
                        0955   613 ;---------------------------------------------------------------------------------
                        0955   614 PCS$MI_ERR:
                        0955   615          $DS_BGNMESSAGE  REGMASK=<R3,R4,R5,R6>
                  0078  0955                        .WORD   ^M<R3,R4,R5,R6>             ; ENTRY MASK
       54   14 AC   D0  0957   616          MOVL    ERR$_P1(AP),R4              ; get address of error
       55   18 AC   D0  095B   617          MOVL    ERR$_P2(AP),R5              ; get expected data
       53   1C AC   D0  095F   618          MOVL    ERR$_P3(AP),R3              ; get actual data
      56    53  55  CD  0963   619          XORL3   R5,R3,R6                    ; get XOR of data
                        0967   620          $DS_PRINTB_S    FORMAT=PCS$T_DPERR,P1=R4,P2=R5,P3=R3,P4=R6
                56  DD  0967                        PUSHL   R6
                53  DD  0969                        PUSHL   R3
                55  DD  096B                        PUSHL   R5
                54  DD  096D                        PUSHL   R4
          FD3E CF   9F  096F                        PUSHAB  PCS$T_DPERR
      000100E0 9F   05  FB  0973                    CALLS   #$$N, @#DS$PRINTB
                        097A   621          $DS_ENDMESSAGE
                    04  097A                        RET                         ; RETURN TO DIAGNOSTIC SUPERVISOR
                        097B   622 ;---------------------------------------------------------------------------------
                        097B   623 PCS$PCS_ERR:
                        097B   624          $DS_BGNMESSAGE  REGMASK=<R3,R4,R5>
                  0038  097B                        .WORD   ^M<R3,R4,R5>                ; ENTRY MASK
            53  0F  D0  097D   625          MOVL    #15,R3                      ; get expected data
      54    55  53  CD  0980   626          XORL3   R3,R5,R4                    ; get XOR of data
                        0984   627          $DS_PRINTB_S    FORMAT=PCS$F_PCS_ERR,P1=R3,P2=R5,P3=R4
                54  DD  0984                        PUSHL   R4
                55  DD  0986                        PUSHL   R5
                53  DD  0988                        PUSHL   R3
```

```
              FECA CF    9F   098A                        PUSHAB   PCS$F_PCS_ERR
      000100E0 9F    04   FB   098E                        CALLS    #$$N, a#DS$PRINTB
                              0995    628          $DS_ENDMESSAGE
                         04   0995                         RET                         ; RETURN TO DIAGNOSTIC SUPERVISOR
                              0996    629 ;----------------------------------------------------------------------------------- ---
                              0996    630 PCS$NO_PARERR_MSG:
                              0996    631          $DS_BGNMESSAGE  REGMASK=<R2,R7> ; Save R2 and R7
                       0084   0996                         .WORD    ^M<R2,R7>                   ; ENTRY MASK
                              0998    632          $DS_PRINTB_S     FORMAT=PCS$T_MC_MESS
              FBF6 CF    9F   0998                         PUSHAB   PCS$T_MC_MESS
      000100E0 9F    01   FB   099C                        CALLS    #$$N, a#DS$PRINTB
                              09A3    633          $DS_PRINTB_S     FORMAT=aERR$_P1(AP)
                14 BC    9F   09A3                         PUSHAB   aERR$_P1(AP)
      000100E0 9F    01   FB   09A6                        CALLS    #$$N, a#DS$PRINTB
                              09AD    634          $DS_PRINTB_S     FORMAT=PCS$T_MC_ERR
              FC34 CF    9F   09AD                         PUSHAB   PCS$T_MC_ERR
      000100E0 9F    01   FB   09B1                        CALLS    #$$N, a#DS$PRINTB
      57   00000000'EF    D0   09B8    635          MOVL     A_MCHK_BUFFER,R7            ; load address of actual data
         67    F84D CF    7D   09BF    636          MOVQ     PCS$TABLE1,(R7)             ; load size and type
                              09C4    637          $DS_PRINTSIG_G  ARGPTR=(R7)
      00010100 9F    67   FA   09C4                        CALLG    (R7), a#DS$PRINTSIG
                              09CB    638          $DS_PRINTB_S     FORMAT=PCS$T_EXPECTED
              FC47 CF    9F   09CB                         PUSHAB   PCS$T_EXPECTED
      000100E0 9F    01   FB   09CF                        CALLS    #$$N, a#DS$PRINTB
                              09D6    639          $DS_PRINTSIG_G  ARGPTR=aERR$_P2(AP)
   00010100 9F    18 BC    FA   09D6                        CALLG    aERR$_P2(AP), a#DS$PRINTSIG
                              09DE    640          $DS_ENDMESSAGE
                         04   09DE                         RET                         ; RETURN TO DIAGNOSTIC SUPERVISOR
                              09DF    641 ;--------------------------------------------------------------------------------------
                              09DF    642
                              09DF    643          $DS_PAGE
```

```
                                  09DF                 .SBTTL  TEST 13:  PCS Memory Test
                              00000000                         .PSECT  TEST_013, PAGE, NOWRT
                                  0014
                                  0014   646 ;++
                                  0014   647 ; Test description:
                                  0014   648 ;
                                  0014   649 ;        This test checks the PCS as a parity memory with standard memory tests.
                                  0014   650 ;        These tests include floating ones and zeros on the data path, testing
                                  0014   651 ;        group select logic , floating ones and zeros on the address lines and
                                  0014   652 ;        testing the RAMs using the moving inversions algorithm.
                                  0014   653 ;
                                  0014   654 ;
                                  0014   655 ;--
                                  0014   656
                                  0014   657          $DS_BGNTEST     <PCS,DEFAULT,ALL>
                                  0014       DATA_013:
                        00000000  0014                 .LONG   0                 ; TEST ARGUMENT TABLE TERMINATOR
                                  0018       TEST_013::
                             0000 0018                 .WORD   ^M<>              ; ENTRY MASK
          54   00000000'8F   DB   001A   658          MFPR    #PR$_SID,R4       ; get system id register
        55   54    08   08   EF   0021   659          EXTZV   #8,#8,R4,R5       ; extract the micro code revision level
         00000000'EF    55   D1   0026   660          CMPL    R5,L_REV_CHECK    ; check the revision level
                        45   1A   002D   661          BGTRU   5$                ; br = this is a L0008 module
         0000FE54'EF    01   D1   002F   662          CMPL    #1,DSA$GL_PASSNO  ; ckeck for first pass
                        03   13   0036   663          BEQLU   1$                ; br = first pass
                      0020   31   0038   664          BRW     2$                ; Not first pass, skip to 2$
  0000FE00'EF   80000000 8F   D3   003B   665 1$:      BITL    #DSA$M_APT,DSA$GL_FLAGS ; Check to see if APT is present
                        19   13   0046   666          BEQLU   3$                ; Branch = No APT
                                  0048   667          $DS_ERRHARD_S   UNIT=#0,MSGADR=PCS$T_NO_PCS
                        00   DD   0048                 PUSHL   #0
               00000623'EF   DF   004A                 PUSHAL  PCS$T_NO_PCS
                        00   DD   0050                 PUSHL   #0
                        01   DD   0052                 PUSHL   #$ER
                                  0054       ;;; TEST 13, SUBTEST 0, ERROR 1
         000100D0 9F    04   FB   0054                 CALLS   #$$M, @#DS$ERRHARD
                                  005B   668                                          ; ERRHARD is for APT, NO L0008 module
                                  005B   669 2$:      $DS_EXIT        TEST
                   50   01   D0   005B                 MOVL    #1,R0             ; NOT AN ERROR
                      03C4   31   005E                 BRW     TEST_013_X        ; EXIT TEST 13
                                  0061   670 3$:      $DS_PRINTF_S    FORMAT=PCS$T_NO_PCS
               00000623'EF   9F   0061                 PUSHAB  PCS$T_NO_PCS
         000100F0 9F    01   FB   0067                 CALLS   #$$N, @#DS$PRINTF
                                  006E   671          $DS_EXIT        TEST              ; not L0008 module, exit the test
                   50   01   D0   006E                 MOVL    #1,R0             ; NOT AN ERROR
                      03B1   31   0071                 BRW     TEST_013_X        ; EXIT TEST 13
          54   00000018'EF   D0   0074   672 5$:      MOVL    PCS$A_PCS_ADDRESS,R4
                                  007B   673          $DS_PROBE_S     ADDRESS=(R4),LENGTH=#4,UNIT=#0
                        00   DD   007B                 PUSHL   #0
                        04   DD   007D                 PUSHL   #4
                        64   9F   007F                 PUSHAB  (R4)
         000101A0 9F    03   FB   0081                 CALLS   #3, @#DS$PROBE
                     42 50   E8   0088   674          BLBS    R0,15$                    ; br = address responded
         0000FE54'EF    01   D1   008B   675          CMPL    #1,DSA$GL_PASSNO          ; ckeck for first pass
                        33   12   0092   676          BNEQ    10$                       ; br = not first pass
  0000FE00'EF   80000000 8F   D3   0094   677          BITL    #DSA$M_APT,DSA$GL_FLAGS ; Check to see if APT is present
```

```
                  19    13  009F    678          BEQLU   8$                    ; Branch = No APT
                            00A1    679          $DS_ERRHARD_S   UNIT=#0,MSGADR=PCS$T_PCS_NORESPOND
                  00    DD  00A1                         PUSHL   #0
          0000064D'EF   DF  00A3                         PUSHAL  PCS$T_PCS_NORESPOND
                  00    DD  00A9                         PUSHL   #0
                  02    DD  00AB                         PUSHL   #$ER
                            00AD        ;;; TEST 13, SUBTEST 0, ERROR 2
      000100D0 9F   04  FB  00AD                         CALLS   #$$M, @#DS$ERRHARD
                            00B4    680                                        ; ERRHARD is for APT, No response
                            00B4    681                                        ; from the PCS module
                            00B4    682          $DS_EXIT        TEST
               50   01  D0  00B4                         MOVL    #1,R0         ; NOT AN ERROR
                  036B  31  00B7                         BRW     TEST_013_X    ; EXIT TEST 13
                            00BA    683 8$:      $DS_PRINTF_S    FORMAT=PCS$T_PCS_NORESPOND
          0000064D'EF   9F  00BA                         PUSHAB  PCS$T_PCS_NORESPOND
      000100F0 9F   01  FB  00C0                         CALLS   #$$N, @#DS$PRINTF
                            00C7    684 10$:     $DS_EXIT        TEST
               50   01  D0  00C7                         MOVL    #1,R0         ; NOT AN ERROR
                  0358  31  00CA                         BRW     TEST_013_X    ; EXIT TEST 13
                            00CD    685 ;15$:    MTPR    #1,#PR$_CADR          ; disable cache  5.2
               25   00  DA  00CD    686 15$:     MTPR    #0,#PR750$_CADR       ; disable cache
                            00D0    687          $DS_CKLOOP      1$
   00010040 9F   FF67 CF  FA  00D0                       CALLG   1$, @#DS$CKLOOP
                            00D9    688
          000002CE'EF   16  00D9    689          JSB     PCS$CLEAR_PCS         ; initialize PCS
                            00DF    690          $DS_PAGE
```

```
                                    00DF                 .SBTTL  SUBTEST 13.1:  Data Path
                                    00DF   693 ;+
                                    00DF   694 ; Subtest description:
                                    00DF   695 ;
                                    00DF   696 ;       The 11/750 micro word is 80 bits wide. When the PCS is addressed as a
                                    00DF   697 ;       memory, the 80 bit micro word is divided into four 1K by 20 bit rows
                                    00DF   698 ;       of words. This subtest checks the data path for stuck at one, stuck at
                                    00DF   699 ;       zero, and shorted data lines. This is accomplished by floating a one
                                    00DF   700 ;       and a zero through the data path for each 20 bit row of RAMs. If an
                                    00DF   701 ;       error is detected it is reported with information describing the nature
                                    00DF   702 ;       of the error. Any errors detected in this subtest must be corrected
                                    00DF   703 ;       befor going on to the next subtest.
                                    00DF   704 ;
                                    00DF   705 ;
                                    00DF   706 ; Subtest steps:
                                    00DF   707 ;
                                    00DF   708 ;       BGNSUBTEST Data Path
                                    00DF   709 ;       :  Initialize flags
                                    00DF   710 ;       :  Enable writing of PCS
                                    00DF   711 ;       :  Initialize data pattern
                                    00DF   712 ;       :  REPEAT
                                    00DF   713 ;       :  :  REPEAT
                                    00DF   714 ;       :  :  :  Write pattern
                                    00DF   715 ;       :  :  :  Read pattern
                                    00DF   716 ;       :  :  :  IF pattern incorrect
                                    00DF   717 ;       :  :  :  THEN
                                    00DF   718 ;       :  :  :  :  Report error
                                    00DF   719 ;       :  :  :  ENDIF
                                    00DF   720 ;       :  :  :  Update pattern
                                    00DF   721 ;       :  :  : UNTIL all patterns done
                                    00DF   722 ;       :  :  ENDREPEAT
                                    00DF   723 ;       :  : UNTIL all rows done
                                    00DF   724 ;       :  ENDREPEAT
                                    00DF   725 ;       ENDSUBTEST Data Path
                                    00DF   726 ;
                                    00DF   727 ;
                                    00DF   728 ; Errors:
                                    00DF   729 ;
                                    00DF   730 ;
                                    00DF   731 ;       If an error occurs, it can be helpful to allow the program to
                                    00DF   732 ;       continue running to collect data from other failures wthin this subtest
                                    00DF   733 ;       to determine the type of failure that is occuring. An example of this
                                    00DF   734 ;       would be to collect a set of error reports which indicate the same bit
                                    00DF   735 ;       error for all four rows. This might indicate that the error is in the
                                    00DF   736 ;       data path and not an individual RAM error.
                                    00DF   737 ;
                                    00DF   738 ;
                                    00DF   739 ;-
                                    00DF   740
                                    00DF   741
                                    00DF   742         $DS_BGNSUB
                                    00DF       T13_S1::
    00010030 9F   00000000'EF   FA  00DF                        CALLG   $S$, @#DS$BGNSUB
                  00000014'EF   D4  00EA   743         CLRL     PCS$L_FLAGS               ; initialize flags
          54      00000018'EF   D0  00F0   744         MOVL     PCS$A_PCS_ADDRESS,R4      ; load first PCS address
```

```
                   0000001C'FF   D4  00F7   745         CLRL    @PCS$A_ENA_ADDRESS      ; Enable the writing of PCS
                            58   D4  00FD   746         CLRL    R8                      ; initialize row counter
              55   00000028'EF   D0  00FF   747 1$:     MOVL    PCS$DATA_PATTERN,R5     ; load all zeros pattern
                            57   D4  0106   748 2$:     CLRL    R7                      ; initialize bit counter
               55   55   FF 8F   9C  0108   749 3$:     ROTL    #-1,R5,R5               ; float pattern across data
             53   55    14   00   EF  010D   750 4$:     EXTZV   #0,#20,R5,R3            ; extract the data to be written
                       64   53   D0  0112   751         MOVL    R3,(R4)                 ; write data
                       56   64   D0  0115   752         MOVL    (R4),R6                 ; read data
             56   56    14   00   EF  0118   753         EXTZV   #0,#20,R6,R6            ; extract the data read
                       56   53   D1  011D   754         CMPL    R3,R6                   ; compare data
                            25   13  0120   755         BEQL    5$                      ; br = data good
                                     0122   756         $DS_ERRHARD_S   UNIT=#0,MSGADR=PCS$T_DATAPATH,PRLINK=PCS$DPERR,-
                                     0122   757                         P1=R4,P2=R3,P3=R6
                            56   DD  0122                       PUSHL   R6
                            53   DD  0124                       PUSHL   R3
                            54   DD  0126                       PUSHL   R4
                   0000088C'EF   DF  0128                       PUSHAL  PCS$DPERR
                   0000068B'EF   DF  012E                       PUSHAL  PCS$T_DATAPATH
                            00   DD  0134                       PUSHL   #0
                            01   DD  0136                       PUSHL   #$ER
                                     0138           ;;; TEST 13, SUBTEST 1, ERROR 1
              000100D0 9F   07   FB  0138                       CALLS   #$$M, @#DS$ERRHARD
                                     013F   758         $DS_CKLOOP      4$                      ; set loop for write and read
           00010040 9F   CB AF   FA  013F                       CALLG   4$, @#DS$CKLOOP
                                     0147   759 ;5$:    AOBLEQ  #19,R7,3$               ; br = not done floating data 5.2
                    BD 57   14   F3  0147   760 5$:     AOBLEQ  #20,R7,3$               ; br = not done floating data
                                     014B   761
           10 00000014'EF   00   E0  014B   762         BBS     #PCS$V_PAT,PCS$L_FLAGS,6$
                                     0153   763                                         ; br = both data patterns done
              55   0000002C'EF   D0  0153   764         MOVL    PCS$DATA_PATTERN+4,R5   ; point R5 to second pattern
              00000014'EF   01   88  015A   765         BISB2   #PCS$M_PAT,PCS$L_FLAGS  ; set data pattern done flag
                            A3   11  0161   766         BRB     2$                      ; go write second pattern
                       54   04   C0  0163   767 6$:     ADDL2   #4,R4                   ; select next row
              00000014'EF   01   CA  0166   768         BICL2   #PCS$M_PAT,PCS$L_FLAGS  ; clear data pattern done flag
                    8E 58   03   F3  016D   769         AOBLEQ  #3,R8,1$                ; br = not done for all rows
                   000002CE'EF   16  0171   770         JSB     PCS$CLEAR_PCS           ; initialize PCS
                                     0177   771         $DS_ENDSUB
                                     0177           T13_S1_X:
     00010038 9F   00000000'EF   FA  0177                       CALLG   $S$, @#DS$ENDSUB
                                     0182   772
                                     0182   773         $DS_PAGE
```

```
                                  0182                .SBTTL  SUBTEST 13.2:  Row Select
                                  0182      ;+
                                  0182      ; Subtest description:
                                  0182      ;
                                  0182      ;         This subtest checks row select (bits 2 and 3 of the address). The PCS
                                  0182      ;         is addressed as four 1K by 20 bit rows of words. A unique pattern is
                                  0182      ;         written to each row, using the addresses that were tested and verrified
                                  0182      ;         in the data path subtest. Then each location is read back to see if the
                                  0182      ;         pattern is correct. If one of the row select lines are stuck one, stuck
                                  0182      ;         zero or shorted together, the data will be over written in at least one
                                  0182      ;         of the four rows. This subtest is dependant on the successful
                                  0182      ;         completion of the data path subtest.
                                  0182      ;
                                  0182      ;
                                  0182      ; Subtest steps:
                                  0182      ;
                                  0182      ;         BNGSUBTEST Row Select
                                  0182      ;         :  Enable writing of PCS
                                  0182      ;         :  Write data patterns
                                  0182      ;         :  Read data patterns
                                  0182      ;         :  IF data patterns are incorrect
                                  0182      ;         :  :  THEN
                                  0182      ;         :  :  Report error
                                  0182      ;         :  ENDIF
                                  0182      ;         ENDSUBTEST Row Select
                                  0182      ;
                                  0182      ;
                                  0182      ; Errors:
                                  0182      ;
                                  0182      ;
                                  0182      ;         When an error is reported, the contents of the four addresses written
                                  0182      ;         and read will be displayed. An example of an error could be the low
                                  0182      ;         order bit of the row select is stuck zero. This means that instead of
                                  0182      ;         selecting row 0, 1, 2, and 3 (each written with a unique pattern), row
                                  0182      ;         0, 1, 0, 1 will be selected. When the data is read back row 0 will have
                                  0182      ;         pattern 3, row 1 will have pattern 4, and rows 3 and 4 will have
                                  0182      ;         correct data. After an error has been corrected in this subtest,
                                  0182      ;         testing must must continue with the data path subtest.
                                  0182      ;
                                  0182      ;-
                                  0182                $DS_BGNSUB
                                  0182      T13_S2::
00010030 9F   0000000C'EF    FA   0182                          CALLG   $S$, a#DS$BGNSUB
              00000014'EF    D4   018D                CLRL      PCS$L_FLAGS                ; initialize flags
              000002B8'EF    16   0193                JSB       PCS$CLEAR_BUFFER           ; zero buffer areas
              00000018'FF    D4   0199                CLRL      aPCS$A_PCS_ADDRESS         ; Enable the writing of PCS
         54   00000018'EF    D0   019F                MOVL      PCS$A_PCS_ADDRESS,R4       ; load first test address
                   58   01   D0   01A6                MOVL      #1,R8                      ; initialize row counter
                   64   58   D0   01A9      1$:       MOVL      R8,(R4)                    ; write data
                   54   04   C0   01AC                ADDL2     #4,R4                      ; update address for next row
                F6 58   04   F3   01AF                AOBLEQ    #4,R8,1$                   ; br = not all rows done
         54   00000018'EF    D0   01B3                MOVL      PCS$A_PCS_ADDRESS,R4       ; load first address to read
                   58   01   D0   01BA                MOVL      #1,R8                      ; initialize row counter
         55   00000030'EF    DE   01BD                MOVAL     PCS$A_PCS_BUF,R5           ; load address of buffer for data
                   53   64   D0   01C4      2$:       MOVL      (R4),R3                    ; read data
         53   53    14    00    EF  01C7   828          EXTZV    #0,#20,R3,R3               ; extract data from word written
                    85    53    D0  01CC   829          MOVL     R3,(R5)+                   ; put data into buffer
                    54    04    C0  01CF   830          ADDL2    #4,R4                      ; update address for next row
                 EE 58    04    F3  01D2   831          AOBLEQ   #4,R8,2$                   ; br = not all rows done
                    58    01    D0  01D6   832          MOVL     #1,R8                      ; initialize data and row counter
           55   00000030'EF     DE  01D9   833          MOVAL    PCS$A_PCS_BUF,R5           ; load address of buffer
                    58    85    D1  01E0   834 3$:      CMPL     (R5)+,R8                   ; check data
                    07    13        01E3   835          BEQL     4$                         ; br = no data error
           00000014'EF    04    C8  01E5   836          BISL2    #PCS$M_ROW_ERR,PCS$L_FLAGS
                                    01EC   837                                             ; set error flag
                 F0 58    04    F3  01EC   838 4$:      AOBLEQ   #4,R8,3$                   ; br = not all rows done
        17 00000014'EF    02    E1  01F0   839          BBC      #PCS$V_ROW_ERR,PCS$L_FLAGS,5$
                                    01F8   840                                             ; br = no error
                                    01F8   841          $DS_ERRHARD_S   UNIT=#0,MSGADR=PCS$T_ROW,PRLINK=PCS$ROWERR
              000008AF'EF       DF  01F8                         PUSHAL   PCS$ROWERR
              000006EA'EF       DF  01FE                         PUSHAL   PCS$T_ROW
                          00    DD  0204                         PUSHL    #0
                          01    DD  0206                         PUSHL    #$ER
                                    0208       ;;; TEST 13, SUBTEST 2, ERROR 1
           000100D0 9F    04    FB  0208                         CALLS    #$$M, @#DS$ERRHARD
                                    020F   842 5$:
              000002CE'EF       16  020F   843          JSB      PCS$CLEAR_PCS              ; initialize PCS
                                    0215   844          $DS_ENDSUB
                                    0215       T13_S2_X:
   00010038 9F    0000000C'EF   FA  0215                         CALLG    $S$, @#DS$ENDSUB
                                    0220   845
                                    0220   846          $DS_PAGE
```

```
                                   0220           .SBTTL  SUBTEST 13.3:  Address Path
                                   0220    849 ;+
                                   0220    850 ; Subtest description:
                                   0220    851 ;
                                   0220    852 ;      This subtest checks the address path for bits stuck at one, stuck at
                                   0220    853 ;      zero and shorted together. The PCS has a ten bit address path which
                                   0220    854 ;      runs in parallel with all of the rams in all four rows. Each row is
                                   0220    855 ;      tested seperatly by writing an encoded incremental pattern to a set of
                                   0220    856 ;      addresses, reading the data back, decoding it and checking
                                   0220    857 ;      for errors. The code used is the Reed - Muller (32,6) code which allows
                                   0220    858 ;      for up to six RAM failures befor detecting an error. This method aides
                                   0220    859 ;      in distingushing RAM errors from address path errors. Floating a one
                                   0220    860 ;      across the address path makes up the set of addresses used in this
                                   0220    861 ;      subtest.
                                   0220    862 ;
                                   0220    863 ;
                                   0220    864 ; Subtest steps:
                                   0220    865 ;
                                   0220    866 ;      BGNSUBTEST Address Path
                                   0220    867 ;        :  Initialize flags
                                   0220    868 ;        :  Enable writing of PCS
                                   0220    869 ;        :  REPEAT
                                   0220    870 ;        :  :  Write encoded data to all addresses
                                   0220    871 ;        :  :  Read encoded data from all addresses
                                   0220    872 ;        :  :  Decode data
                                   0220    873 ;        :  :  Check data
                                   0220    874 ;        :  :  IF any data errors
                                   0220    875 ;        :  :  :  THEN
                                   0220    876 ;        :  :  :  Report data path error
                                   0220    877 ;        :  :  ENDIF
                                   0220    878 ;        :  : UNTIL all four rows done
                                   0220    879 ;        :  ENDREPEAT
                                   0220    880 ;      ENDSUBTEST Address Path
                                   0220    881 ;
                                   0220    882 ;
                                   0220    883 ;-
                                   0220    884
                                   0220    885
                                   0220    886            $DS_BGNSUB
                                   0220        T13_S3::
 00010030 9F   00000018'EF    FA   0220                   CALLG   $S$, @#DS$BGNSUB
               00000014'EF    D4   022B    887    CLRL    PCS$L_FLAGS             ; init flags
                         54   D4   0231    888    CLRL    R4                      ; initialize row counter
               0000001C'FF    D4   0233    889    CLRL    @PCS$A_ENA_ADDRESS      ; Enable writing of PCS
          58   00000018'EF    D0   0239    890    MOVL    PCS$A_PCS_ADDRESS,R8    ; load first PCS address
                    55   58   D0   0240    891 1$:  MOVL    R8,R5                 ; load it in R5 too
                    53   08   D0   0243    892    MOVL    #8,R3                   ; initialize floating address
          56   000001C0'EF    DE   0246    893    MOVAL   PCS$CODE_WORDS,R6       ; load address of encoded data
          57   00000130'EF    DE   024D    894    MOVAL   PCS$NOISY_WORDS,R7      ; load address of data buffer
                    65   86   D0   0254    895 2$:  MOVL    (R6)+,(R5)            ; write test data
                    87   65   D0   0257    896    MOVL    (R5),(R7)+              ; read test data
               53   53   01   9C   025A    897    ROTL    #1,R3,R3                ; float one across address
               55   58   53   C1   025E    898    ADDL3   R3,R8,R5                ; build next test address
                 EE 53   0E   E1   0262    899    BBC     #PCS$V_BIT14,R3,2$      ; br = not finished
               00000303'EF    16   0266    900    JSB     PCS$RM_DECODE           ; go decode test data
```

```
                             59  D4  026C   901          CLRL    R9                      ; init counter
              5A    000000B0'EF  DE  026E   902          MOVAL   PCS$DECODED_WORDS,R10   ; load address of decoded words
                       8A    59  D1  0275   903 3$:      CMPL    R9,(R10)+               ; check data
                             07  13  0278   904          BEQL    4$                      ; br = no error
              00000014'EF    02  C8  027A   905          BISL2   #PCS$M_DATA_ERR,PCS$L_FLAGS
                                     0281   906                                          ; set error flag
                    F0 59    0A  F3  0281   907 4$:      AOBLEQ  #10,R9,3$               ; br = not finished checking data
           19 00000014'EF    01  E1  0285   908          BBC     #PCS$V_DATA_ERR,PCS$L_FLAGS,5$
                                     028D   909                                          ; br = no error
                                     028D   910          $DS_ERRHARD_S   UNIT=#0,MSGADR=PCS$T_ADR_ERR,PRLINK=PCS$ADDR_ERR,P1=R8
                             58  DD  028D                        PUSHL   R8
                    000008FF'EF  DF  028F                        PUSHAL  PCS$ADDR_ERR
                    0000076C'EF  DF  0295                        PUSHAL  PCS$T_ADR_ERR
                             00  DD  029B                        PUSHL   #0
                             01  DD  029D                        PUSHL   #$ER
                                     029F       ;;; TEST 13, SUBTEST 3, ERROR 1
              000100D0 9F    05  FB  029F                        CALLS   #$$M, @#DS$ERRHARD
                       58    04  C0  02A6   911 5$:      ADDL2   #4,R8                   ; update row select
                    93 54    03  F3  02A9   912          AOBLEQ  #3,R4,1$                ; br = not all rows done
                    000002CE'EF  16  02AD   913          JSB     PCS$CLEAR_PCS           ; initialize PCS
                                     02B3   914
                                     02B3   915          $DS_ENDSUB
                                     02B3       T13_S3_X:
     00010038 9F    00000018'EF  FA  02B3                        CALLG   $S$, @#DS$ENDSUB
                                     02BE   916
                                     02BE   917          $DS_PAGE
```

```
02BE                  .SBTTL  SUBTEST 13.4:  Moving Inversions
02BE        ;+
02BE        ; Subtest description:
02BE        ;
02BE        ;         This subtest does an exhaustive test of all RAM chips using the
02BE        ;         moving inversions algorithem. Each address accessed in this subtest
02BE        ;         will be read, written and then read again with data which is checked
02BE        ;         after each read. This subtest starts at the first address and works
02BE        ;         its way up thru PCS memory, eventually writing all locations with all
02BE        ;         ones. Then the process is started again and is complete when all
02BE        ;         locations have been written with all zeros. Next this is done working
02BE        ;         from the last address to the first, both writting till the data is all
02BE        ;         ones and then all zeros. This is done for all bit positions in all RAMS
02BE        ;         (four bits). For each successive pass a different address bit is used
02BE        ;         as the lease signifigant bit in counting such that every address line
02BE        ;         is toggled at the fastest possible rate.
02BE        ;
02BE        ;
02BE        ; Subtest steps:
02BE        ;
02BE        ;         BGNSUBTEST Moving Inversions
02BE        ;         :  REPEAT
02BE        ;         :  :  Set increment to 1
02BE        ;         :  :  REPEAT
02BE        ;         :  :  :  Initialize flags
02BE        ;         :  :  :  Point to first entry in data table
02BE        ;         :  :  :  Set forward direction
02BE        ;         :  :  :  REPEAT
02BE        ;         :  :  :  :  REPEAT
02BE        ;         :  :  :  :  :  Set address to starting address
02BE        ;         :  :  :  :  :  REPEAT
02BE        ;         :  :  :  :  :  :  Read data
02BE        ;         :  :  :  :  :  :  IF data error
02BE        ;         :  :  :  :  :  :  :  THEN
02BE        ;         :  :  :  :  :  :  :  Report data error
02BE        ;         :  :  :  :  :  :  ENDIF
02BE        ;         :  :  :  :  :  :  Write data
02BE        ;         :  :  :  :  :  :  Read data
02BE        ;         :  :  :  :  :  :  IF data error
02BE        ;         :  :  :  :  :  :  :  THEN
02BE        ;         :  :  :  :  :  :  :  Report data error
02BE        ;         :  :  :  :  :  :  ENDIF
02BE        ;         :  :  :  :  :  :  Update address
02BE        ;         :  :  :  :  :  : UNTIL all addresses done
02BE        ;         :  :  :  :  :  ENDREPEAT
02BE        ;         :  :  :  :  :  Update data
02BE        ;         :  :  :  :  : UNTIL all data done
02BE        ;         :  :  :  :  ENDREPEAT
02BE        ;         :  :  :  :  Change direction
02BE        ;         :  :  :  : UNTIL reverse is done
02BE        ;         :  :  :  ENDREPEAT
02BE        ;         :  :  :  Update row
02BE        ;         :  :  : UNTIL all rows done
02BE        ;         :  :  ENDREPEAT
02BE        ;         :  :  Update increment
```

```
                                     02BE   974 ;       :  : UNITL increment = MSB of address
                                     02BE   975 ;       :  ENDREPEAT
                                     02BE   976 ;       ENDSUBTEST Moving Inversions
                                     02BE   977 ;
                                     02BE   978 ;
                                     02BE   979 ;-
                                     02BE   980
                                     02BE   981
                                     02BE   982         $DS_BGNSUB
                                     02BE       T13_S4::
     00010030 9F    00000024'EF   FA  02BE                       CALLG   $S$, @#DS$BGNSUB
     0000000C'EF    000003EE'EF   DE  02C9   983         MOVAL   PCS$RENTRY1,PCS$A_RETURN; load rentry address for unexpected MC
                                     02D4   984         $DS_SETVEC_S    VECTOR=#SCB$L_MACHCHK,SRVADR=PCS$CHECK_MCHK
                            00   DD  02D4                       PUSHL   #0
                   000004B8'EF   DF  02D6                       PUSHAL  PCS$CHECK_MCHK
                            04   DD  02DC                       PUSHL   #SCB$L_MACHCHK
              00010160 9F   03   FB  02DE                       CALLS   #3, @#DS$SETVEC
                                     02E5   985 PCS$TAG1:
                   00000014'EF   D4  02E5   986         CLRL    PCS$L_FLAGS             ; initialize flags
                   00000008'EF   D4  02EB   987         CLRL    PCS$L_ROW               ; start with row 0
              00000004'EF   10   D0  02F1   988 5$:     MOVL    #16,PCS$L_INC           ; initialize address increment
                   000003C4'EF   16  02F8   989 10$:    JSB     PCS$SETUP_ADDR          ; go set up address table for this inc
              53   000001EC'EF   DE  02FE   990 20$:    MOVAL   PCS$DATA_TABLE,R3       ; setup pointer to read data
              54   000001F0'EF   DE  0305   991         MOVAL   PCS$DATA_TABLE+4,R4     ; setup pointer to write data
           09 00000014'EF   03   E1  030C   992 30$:    BBC     #PCS$V_REVERSE,PCS$L_FLAGS,40$
                                     0314   993                                         ; br = doing forward
              52   000003FF 8F   D0  0314   994         MOVL    #1023,R2                ; point offset to last address in table
                            02   11  031B   995         BRB     50$                     ; skip
                            52   D4  031D   996 40$:    CLRL    R2                      ; point offset to first address in table
              5B   00000000'EF   D0  031F   997 50$:    MOVL    PCS$A_PCS_BUFFER,R11    ; get starting address of addresses
                     5A   6B42   D0  0326   998         MOVL    (R11)[R2],R10           ; load address of data
                       55   6A   D0  032A   999         MOVL    (R10),R5                ; read data
             55   55   14   00   EF  032D  1000         EXTZV   #0,#20,R5,R5            ; extract data from longword
                       63   55   D1  0332  1001         CMPL    R5,(R3)                 ; check data read
                            1D   13  0335  1002         BEQL    60$                     ; br = data good
                                     0337  1003         $DS_ERRHARD_S   UNIT=#0,MSGADR=PCS$T_MI_ERR1,PRLINK=PCS$MI_ERR,-
                                     0337  1004                         P1=R10,P2=R5,P3=(R3)
                            63   DD  0337                       PUSHL   (R3)
                            55   DD  0339                       PUSHL   R5
                            5A   DD  033B                       PUSHL   R10
                   00000955'EF   DF  033D                       PUSHAL  PCS$MI_ERR
                   0000079A'EF   DF  0343                       PUSHAL  PCS$T_MI_ERR1
                            00   DD  0349                       PUSHL   #0
                            01   DD  034B                       PUSHL   #$ER
                                     034D       ;;; TEST 13, SUBTEST 4, ERROR 1
              000100D0 9F   07   FB  034D                       CALLS   #$$M, @#DS$ERRHARD
                       6A   64   D0  0354  1005 60$:    MOVL    (R4),(R10)              ; write data
                       55   6A   D0  0357  1006         MOVL    (R10),R5                ; read data
             55   55   14   00   EF  035A  1007         EXTZV   #0,#20,R5,R5            ; extract data from longword
                       64   55   D1  035F  1008         CMPL    R5,(R4)                 ; check data read
                            1D   13  0362  1009         BEQL    70$                     ; br = data good
                                     0364  1010         $DS_ERRHARD_S   UNIT=#0,MSGADR=PCS$T_MI_ERR2,PRLINK=PCS$MI_ERR,-
                                     0364  1011                         P1=R10,P2=R5,P3=(R4)
                            64   DD  0364                       PUSHL   (R4)
                            55   DD  0366                       PUSHL   R5
                            5A   DD  0368                       PUSHL   R10
                   00000955'EF   DF  036A                       PUSHAL  PCS$MI_ERR
                   000007D0'EF   DF  0370                       PUSHAL  PCS$T_MI_ERR2
                            00   DD  0376                       PUSHL   #0
                            02   DD  0378                       PUSHL   #$ER
                                     037A       ;;; TEST 13, SUBTEST 4, ERROR 2
              000100D0 9F   07   FB  037A                       CALLS   #$$M, @#DS$ERRHARD
                                     0381  1012 70$:    $DS_BREAK
              00010058 9F   6E   FA  0381                       CALLG   (SP), @#DS$BREAK
           07 00000014'EF   03   E1  0388  1013         BBC     #PCS$V_REVERSE,PCS$L_FLAGS,80$
                                     0390  1014                                         ; br = forward direction
                       52   01   C2  0390  1015         SUBL2   #1,R2                   ; index to next address (reverse)
                            8A   18  0393  1016         BGEQ    50$                     ; br = not finished
                            0F   11  0395  1017         BRB     100$                    ; skip
                       52   01   C0  0397  1018 80$:    ADDL2   #1,R2                   ; index to next address (forward)
              000003FF 8F   52   D1  039A  1019         CMPL    R2,#1023                ; finished forward
                            03   14  03A1  1020         BGTR    100$                    ; br = finished
                          FF79   31  03A3  1021         BRW     50$                     ; not finished
                       84   83   D1  03A6  1022 100$:   CMPL    (R3)+,(R4)+             ; update data
                            63   D5  03A9  1023         TSTL    (R3)                    ; check for data done
                            03   13  03AB  1024         BEQL    105$                    ; br = do reverse
                          FF5C   31  03AD  1025         BRW     30$                     ; br = do forward for next data
           0A 00000014'EF   03   E0  03B0  1026 105$:   BBS     #PCS$V_REVERSE,PCS$L_FLAGS,108$
                                     03B8  1027                                         ; br = reverse done
              00000014'EF   08   C8  03B8  1028         BISL2   #PCS$M_REVERSE,PCS$L_FLAGS
                                     03BF  1029                                         ; set reverse flag
                          FF3C   31  03BF  1030         BRW     20$                     ; do reverse
00000004'EF   00000004'EF   01   9C  03C2  1031 108$:   ROTL    #1,PCS$L_INC,PCS$L_INC  ; shift increment
           03 00000004'EF   0E   E0  03CE  1032         BBS     #PCS$V_BIT14,PCS$L_INC,109$
                                     03D6  1033                                         ; br = all increments done
                          FF1F   31  03D6  1034         BRW     10$                     ; go do another increment
              00000008'EF   04   C0  03D9  1035 109$:   ADDL2   #4,PCS$L_ROW            ; update row to test
                                     03E0  1036 ;       CMPL    PCS$L_ROW,#8            ; check for finished 5.2
              0C   00000008'EF   D1  03E0  1037         CMPL    PCS$L_ROW,#12           ; check for finished
                            03   14  03E7  1038         BGTR    110$                    ; br = finished
                          FF05   31  03E9  1039         BRW     5$                      ; br = not finished
                            1C   11  03EC  1040 110$:   BRB     PCS$TAG2
                                     03EE  1041 PCS$RENTRY1:
                                     03EE  1042         $DS_ERRHARD_S   UNIT=#0,MSGADR=PCS$T_CS_PARITY1
                            00   DD  03EE                       PUSHL   #0
                   000904FF'EF   DF  03F0                       PUSHAL  PCS$T_CS_PARITY1
                            00   DD  03F6                       PUSHL   #0
                            03   DD  03F8                       PUSHL   #$ER
                                     03FA       ;;; TEST 13, SUBTEST 4, ERROR 3
              000100D0 9F   04   FB  03FA                       CALLS   #$$M, @#DS$ERRHARD
                                     0401  1043         $DS_CKLOOP      PCS$TAG1
         00010040 9F   FEE0 CF   FA  0401                       CALLG   PCS$TAG1, @#DS$CKLOOP
                                     040A  1044 PCS$TAG2:
                                     040A  1045         $DS_CLRVEC_S    VECTOR=SCB$L_MACHCHK
                   00000004'EF   DD  040A                       PUSHL   SCB$L_MACHCHK
              00010168 9F   01   FB  0410                       CALLS   #1, @#DS$CLRVEC
                                     0417  1046         $DS_ENDSUB
                                     0417       T13_S4_X:
     00010038 9F   00000024'EF   FA  0417                       CALLG   $S$, @#DS$ENDSUB
                                     0422  1047
```

```
                                      0422  1048          $DS_ENDTEST
                        50    01   D0 0422                       MOVL    #1, R0          ; NORMAL EXIT
                                      0425      TEST_013_X::
              00010058 9F    6E   FA 0425                       CALLG   (SP), @#DS$BREAK
                                   04 042C                       RET                     ; RETURN TO TEST SEQUENCER
                                      042D  1049
                                      042D  1050          $DS_PAGE
                                   042D              .SBTTL  TEST 14:  PCS Execution Test
                               00000000                      .PSECT  TEST_014, PAGE, NOWRT
                                   0018
                                   0018  1053 ;++
                                   0018  1054 ; Test description:
                                   0018  1055 ;
                                   0018  1056 ;          This test checks that micro code can be executed out of patch
                                   0018  1057 ;          control store.
                                   0018  1058 ;
                                   0018  1059 ;--
                                   0018  1060
                                   0018  1061          $DS_BGNTEST       <PCS,DEFAULT,ALL>
                                   0018       DATA_014:
                       00000000    0018                      .LONG    0                  ; TEST ARGUMENT TABLE TERMINATOR
                                   001C       TEST_014::
                           0000    001C                      .WORD    ^M<>               ; ENTRY MASK
          54   00000000'8F    DB   001E  1062          MFPR    #PR$_SID,R4                 ; get system id register
       55    54    08    08    EF   0025  1063          EXTZV   #8,#8,R4,R5                 ; extract the micro code revision level
        00000000'EF    55    D1   002A  1064          CMPL    R5,L_REV_CHECK              ; check the revision level
                       45    1A   0031  1065          BGTRU   5$                          ; br = this is a L0008 modu e
        0000FE54'EF    01    D1   0033  1066          CMPL    #1,DSA$GL_PASSNO            ; ckeck for first pass
                       03    13   003A  1067          BEQLU   1$                          ; br = first pass
                     0020    31   003C  1068          BRW     2$                          ; Not first pass, skip to 2$
   0000FE00'EF    80000000 8F    D3   003F  1069 1$:    BITL    #DSA$M_APT,DSA$GL_FLAGS ; Check to see if APT is present
                       19    13   004A  1070          BEQLU   3$                          ; Branch = No APT
                                   004C  1071          $DS_ERRHARD_S    UNIT=#0,MSGADR=PCS$T_NO_PCS
                       00    DD   004C                      PUSHL    #0
           00000623'EF    DF   004E                      PUSHAL   PCS$T_NO_PCS
                       00    DD   0054                      PUSHL    #0
                       01    DD   0056                      PUSHL    #$ER
                                   0058       ;;; TEST 14, SUBTEST 0, ERROR 1
        000100D0 9F    04    FB   0058                      CALLS    #$$M, @#DS$ERRHARD
                                   005F  1072                                                  ; ERRHARD is for APT, NO L0008 module
                                   005F  1073 2$:     $DS_EXIT           TEST
               50    01    D0   005F                      MOVL     #1,R0              ; NOT AN ERROR
                     02C3    31   0062                      BRW      TEST_014_X         ; EXIT TEST 14
                                   0065  1074 3$:     $DS_PRINTF_S       FORMAT=PCS$T_NO_PCS
           00000623'EF    9F   0065                      PUSHAB   PCS$T_NO_PCS
        000100F0 9F    01    FB   006B                      CALLS    #$$N, @#DS$PRINTF
                                   0072  1075          $DS_EXIT           TEST               ; not L0008 module, exit the test
               50    01    D0   0072                      MOVL     #1,R0              ; NOT AN ERROR
                     02B0    31   0075                      BRW      TEST_014_X         ; EXIT TEST 14
          54   00000018'EF    D0   0078  1076 5$:    MOVL    PCS$A_PCS_ADDRESS,R4
                                   007F  1077          $DS_PROBE_S        ADDRESS=(R4),LENGTH=#4,UNIT=#0
                       00    DD   007F                      PUSHL    #0
                       04    DD   0081                      PUSHL    #4
                       64    9F   0083                      PUSHAB   (R4)
        000101A0 9F    03    FB   0085                      CALLS    #3, @#DS$PROBE
                    42 50    E8   008C  1078          BLBS    R0,15$                           ; br = address responded
        0000FE54'EF    01    D1   008F  1079          CMPL    #1,DSA$GL_PASSNO                 ; ckeck for first pass
                       33    12   0096  1080          BNEQ    10$                              ; br = not first pass
   0000FE00'EF    80000000 8F    D3   0098  1081          BITL    #DSA$M_APT,DSA$GL_FLAGS ; Check to see if APT is present
                       19    13   00A3  1082          BEQLU   8$                               ; Branch = No APT
                                   00A5  1083          $DS_ERRHARD_S    UNIT=#0,MSGADR=PCS$T_PCS_NORESPOND
                       00    DD   00A5                      PUSHL    #0
          0000064D'EF    DF   00A7                          PUSHAL  PCS$T_PCS_NORESPOND
                   00    DD   00AD                          PUSHL   #0
                   02    DD   00AF                          PUSHL   #$ER
                              00B1        ;;; TEST 14, SUBTEST 0, ERROR 2
      000100D0 9F  04    FB   00B1                          CALLS   #$$M, @#DS$ERRHARD
                              00B8  1084                                              ; ERRHARD is for APT, No response
                              00B8  1085                                              ; from the PCS module
                              00B8  1086            $DS_EXIT        TEST
              50   01    D0   00B8                          MOVL    #1,R0             ; NOT AN ERROR
                 026A    31   00BB                          BRW     TEST_014_X        ; EXIT TEST 14
                              00BE  1087 8$:        $DS_PRINTF_S    FORMAT=PCS$T_PCS_NORESPOND
          0000064D'EF    9F   00BE                          PUSHAB  PCS$T_PCS_NORESPOND
      000100F0 9F  01    FB   00C4                          CALLS   #$$N, @#DS$PRINTF
                              00CB  1088 10$:       $DS_EXIT        TEST
              50   01    D0   00CB                          MOVL    #1,R0             ; NOT AN ERROR
                 0257    31   00CE                          BRW     TEST_014_X        ; EXIT TEST 14
              25   01    DA   00D1  1089 15$:       MTPR    #1,#PR750$_CADR           ; disable cache
                              00D4  1090            $DS_SETVEC_S    VECTOR=#SCB$L_MACHCHK,SRVADR=PCS$CHECK_MCHK
                   00    DD   00D4                          PUSHL   #0
          000004B8'EF    DF   00D6                          PUSHAL  PCS$CHECK_MCHK
                   04    DD   00DC                          PUSHL   #SCB$L_MACHCHK
      00010160 9F  03    FB   00DE                          CALLS   #3, @#DS$SETVEC
          000002CE'EF    16   00E5  1091            JSB     PCS$CLEAR_PCS                 ; initialize PCS
                              00EB  1092            $DS_PAGE
                              00EB  1093
                              00EB  1094
```

```
                                      00EB            .SBTTL  SUBTEST 14.1:  PCS Entry/Exit
                                      00EB  1097 ;+
                                      00EB  1098 ; Subtest description:
                                      00EB  1099 ;
                                      00EB  1100 ;       This subtest attempts to execute from PCS, an instruction which will
                                      00EB  1101 ;       load the  D" register with an initial value and then decrement the
                                      00EB  1102 ;       value and write it into general purpose register R5. The NOP
                                      00EB  1103 ;       instruction is used to execute from PCS and then the register is
                                      00EB  1104 ;       checked for the correct results.
                                      00EB  1105 ;
                                      00EB  1106 ; Subtest steps:
                                      00EB  1107 ;
                                      00EB  1108 ;       BGNSUBTEST PCS Entry/Exit
                                      00EB  1109 ;       :  Disable PCS
                                      00EB  1110 ;       :  Initialize flags
                                      00EB  1111 ;       :  Initialize register
                                      00EB  1112 ;       :  Write micro words into PCS
                                      00EB  1113 ;       :  Execute NOP to enter PCS
                                      00EB  1114 ;       :  Check register
                                      00EB  1115 ;       :  IF Register contents not correct
                                      00EB  1116 ;       :  :  THEN
                                      00EB  1117 ;       :  :  Report error
                                      00EB  1118 ;       :  ENDIF
                                      00EB  1119 ;       ENDSUBTEST PCS Entry/Exit
                                      00EB  1120 ;
                                      00EB  1121 ;
                                      00EB  1122 ;
                                      00EB  1123 ;-
                                      00EB  1124
                                      00EB  1125         $DS_BGNSUB
                                      00EB       T14_S1::
     00010030 9F   00000030'EF    FA  00EB                       CALLG   $S$, a#DS$BGNSUB
                   00000014'EF    D4  00F6  1126         CLRL    PCS$L_FLAGS              ; initialize flags
                   00000000'EF    D4  00FC  1127         CLRL    PCS$L_COUNT              ; setup count for this subtest
     0000000C'EF   00000151'EF    DE  0102  1128         MOVAL   PCS$RENTRY2,PCS$A_RETURN; load rentry address for unexpected MC
                                      010D  1129 PCS$TAG3:
                         55   10  D0  010D  1130         MOVL    #16,R5                   ; initialize test register
                0000001C'FF   01  D0  0110  1131         MOVL    #1,aPCS$A_ENA_ADDRESS    ; Enable writing of patch bit
             57    00000020'EF    D0  0117  1132         MOVL    PCS$A_PTH_LOCATION,R7    ; load address of patch bit to enable
                              67  D4  011E  1133         CLRL    (R7)                     ; Clear (enable) the patch bit
                   0000001C'FF    D4  0120  1134         CLRL    aPCS$A_ENA_ADDRESS       ; Disable writing of patch bit
                   000003FC'EF    16  0126  1135         JSB     PCS$LOAD_MICRO           ; load pcs with micro code and enable
                                  01  012C  1136         NOP                              ; The microword that perfroms the NOP
                                      012D  1137                                          ; has the patch bit set with a next
                                      012D  1138                                          ; address of 0002 which will cause a
                                      012D  1139                                          ; trap to PCS 2802.
                   00000018'FF    D4  012D  1140         CLRL    aPCS$A_PCS_ADDRESS       ; disable PCS
                         0F   55  D1  0133  1141         CMPL    R5,#15                   ; check that register decremented twice
                              45  13  0136  1142         BEQL    PCS$TAG4                 ; br = data good
                                      0138  1143         $DS_ERRHARD_S   UNIT=#0,MSGADR=PCS$T_ENTRY_ERR,PRLINK=PCS$PCS_ERR
                   0000097B'EF    DF  0138                       PUSHAL  PCS$PCS_ERR
                   00000805'EF    DF  013E                       PUSHAL  PCS$T_ENTRY_ERR
                              00  DD  0144                       PUSHL   #0
                              01  DD  0146                       PUSHL   #$ER
                                      0148       ;;; TEST 14, SUBTEST 1, ERROR 1
```

```
        000100D0 9F    04   FB  0148                         CALLS    #$$M, a#DS$ERRHARD
                       2C   11  014F  1144           BRB     PCS$TAG4
                                0151  1145 PCS$RENTRY2:
   00000018'FF    000186A0 8F   CA  0151  1146           BICL2   #00100000,aPCS$A_PCS_ADDRESS
                                015C  1147           $DS_ERRHARD_S    UNIT=#0,MSGADR=PCS$T_CS_PARITY2
                       00   DD  015C                         PUSHL    #0
              00000532'EF   DF  015E                         PUSHAL   PCS$T_CS_PARITY2
                       00   DD  0164                         PUSHL    #0
                       02   DD  0166                         PUSHL    #$ER
                                0168           ;;; TEST 14, SUBTEST 1, ERROR 2
        000100D0 9F    04   FB  0168                         CALLS    #$$M, a#DS$ERRHARD
              000002CE'EF   16  016F  1148           JSB     PCS$CLEAR_PCS
                                0175  1149           $DS_CKLOOP       PCS$TAG3
     00010040 9F     95 AF  FA  0175                         CALLG    PCS$TAG3, a#DS$CKLOOP
                                017D  1150 PCS$TAG4:
              000002CE'EF   16  017D  1151           JSB     PCS$CLEAR_PCS             ; initialize PCS
                                0183  1152           $DS_ENDSUB
                                0183           T14_S1_X:
   00010038 9F    00000030'EF   FA  0183                         CALLG    $S$, a#DS$ENDSUB
                                018E  1153
                                018E  1154           $DS_PAGE
                                018E  1155
                                018E  1156
```

```
                                    018E                 .SBTTL  SUBTEST 14.2:  PCS  Next Field  Addressing
                                    018E   1159 ;+
                                    018E   1160 ; Subtest description:
                                    018E   1161 ;
                                    018E   1162 ;        This subtest checks the PCS `next field  portion of the micro word to
                                    018E   1163 ;        insure that each bit produces the proper transfer address. This is
                                    018E   1164 ;        accomplished by filling PCS memory with micro words that decrement
                                    018E   1165 ;        general purpose register R5 and have a  next field  which points to
                                    018E   1166 ;        the next PCS location. This subtest is an extened version of the
                                    018E   1167 ;        previous subtest and runs thru all of PCS memory. An NOP instruction
                                    018E   1168 ;        is executed to execute from PCS and afterwards R5 is checked to have
                                    018E   1169 ;        the expected value.
                                    018E   1170 ;
                                    018E   1171 ; Subtest steps:
                                    018E   1172 ;
                                    018E   1173 ;        BGNSUBTEST PCS  Next Field  Addressing
                                    018E   1174 ;        :  Disable PCS
                                    018E   1175 ;        :  Initialize flags
                                    018E   1176 ;        :  Initialize register
                                    018E   1177 ;        :  Write PCS with micro words for all PCS
                                    018E   1178 ;        :  Execute NOP instruction
                                    018E   1179 ;        :  Check register to have decremented properly
                                    018E   1180 ;        :  IF register contains wrong data
                                    018E   1181 ;        :  :  THEN
                                    018E   1182 ;        :  :  Report error
                                    018E   1183 ;        :  ENDIF
                                    018E   1184 ;        ENDSUBTEST PCS  Next Field  Addressing
                                    018E   1185 ;
                                    018E   1186 ;--
                                    018E   1187
                                    018E   1188          $DS_BGNSUB
                                    018E        T14_S2::
 00010030 9F   0000003C'EF    FA    018E                       CALLG   $S$, @#DS$BGNSUB
               00000014'EF    D4    0199   1189          CLRL    PCS$L_FLAGS              ; initialize flags
 00000000'EF   000003E5 8F    D0    019F   1190          MOVL    #997,PCS$L_COUNT         ; setup count for all locations
 0000000C'EF   000001FD'EF    DE    01AA   1191          MOVAL   PCS$RENTRY3,PCS$A_RETURN; load rentry address for unexpected MC
                                    01B5   1192 PCS$TAG5:
          55   000003F5 8F    D0    01B5   1193          MOVL    #1013,R5                 ; initialize test register
        0000001C'FF     01    D0    01BC   1194          MOVL    #1,@PCS$A_ENA_ADDRESS    ; Enable writing of patch bit
          57   00000020'EF    D0    01C3   1195          MOVL    PCS$A_PTH_LOCATION,R7    ; load address of patch bit to enable
                         67    D4    01CA   1196          CLRL    (R7)                     ; Clear (enable) the patch bit
               0000001C'FF    D4    01CC   1197          CLRL    @PCS$A_ENA_ADDRESS       ; Disable writing of patch bit
               000003FC'EF    16    01D2   1198          JSB     PCS$LOAD_MICRO           ; load pcs with micro code
                         01          01D8   1199          NOP                              ; The microword that perfroms the NOP
                                    01D9   1200                                           ; has the patch bit set with a next
                                    01D9   1201                                           ; address of 0002 which will cause a
                                    01D9   1202                                           ; trap to PCS 2802.
               00000018'FF    D4    01D9   1203          CLRL    @PCS$A_PCS_ADDRESS       ; disable PCS
                    0E   55    D1    01DF   1204          CMPL    R5,#14                   ; check that register decremented
                         45    13    01E2   1205          BEQL    PCS$TAG6                 ; br = data good
                                    01E4   1206          $DS_ERRHARD_S   UNIT=#0,MSGADR=PCS$T_NXFIELD_ERR,PRLINK-PCS$PCS_ERR
               0000097B'EF    DF    01E4                       PUSHAL  PCS$PCS_ERR
               0000082A'EF    DF    01EA                       PUSHAL  PCS$T_NXFIELD ERR
                         00    DD    01F0                       PUSHL   #0
                         01    DD    01F2                       PUSHL   #$ER
```

```
                                     01F4         ;;; TEST 14, SUBTEST 2, ERROR 1
           000100D0 9F      04   FB  01F4                         CALLS    #$$M, @#DS$ERRHARD
                            2C   11  01FB   1207          BRB     PCS$TAG6
                                     01FD   1208 PCS$RENTRY3:
    00000018'FF     000186A0 8F   CA  01FD   1209          BICL2   #00100000,@PCS$A_PCS_ADDRESS
                                     0208   1210          $DS_ERRHARD_S   UNIT=#0,MSGADR=PCS$T_CS_PARITY3
                            00   DD  0208                         PUSHL    #0
                  0000055E'EF    DF  020A                         PUSHAL   PCS$T_CS_PARITY3
                            00   DD  0210                         PUSHL    #0
                            02   DD  0212                         PUSHL    #$ER
                                     0214         ;;; TEST 14, SUBTEST 2, ERROR 2
           000100D0 9F      04   FB  0214                         CALLS    #$$M, @#DS$ERRHARD
                  000002CE'EF    16  021B   1211          JSB     PCS$CLEAR_PCS
                                     0221   1212          $DS_CKLOOP      PCS$TAG5
       00010040 9F      91 AF    FA  0221                         CALLG    PCS$TAG5, @#DS$CKLOOP
                                     0229   1213 PCS$TAG6:
                  000002CE'EF    16  0229   1214          JSB     PCS$CLEAR_PCS            ; initialize PCS
                                     022F   1215          $DS_ENDSUB
                                     022F        T14_S2_X:
    00010038 9F     0000003C'EF   FA  022F                         CALLG    $S$, @#DS$ENDSUB
                                     023A   1216
```

```
                                          023A  1218         $DS_SBTTL       <PCS Parity Error ABORT>
                                          023A               .SBTTL  SUBTEST 14.3:  PCS Parity Error ABORT
                                          023A  1219 ;+
                                          023A  1220 ; Subtest description:
                                          023A  1221 ;
                                          023A  1222 ;       This subtest attempts to execute from PCS, an instruction which will
                                          023A  1223 ;       attempt to load the  D` register with an initial value and then
                                          023A  1224 ;       decrement the value and write it into general purpose register R5.
                                          023A  1225 ;       The NOP instruction is used to transfer control to PCS.  Entry into
                                          023A  1226 ;       PCS is at 2802 (where D_R[R5],NEXT/2800 is executed).  Then, at
                                          023A  1227 ;       location 2800 ( where D_D-1,NEXT/2801 is executed) a parity error
                                          023A  1228 ;       should be detected.
                                          023A  1229 ;
                                          023A  1230 ; Subtest steps:
                                          023A  1231 ;
                                          023A  1232 ;       BGNSUBTEST PCS Entry/Exit
                                          023A  1233 ;       :  Disable PCS
                                          023A  1234 ;       :  Initialize flags
                                          023A  1235 ;       :  Write micro words into PCS
                                          023A  1236 ;       :  Set-up Return addresses
                                          023A  1237 ;       :  Set the appropriate Patch Bit
                                          023A  1238 ;       :  Execute NOP to enter PCS
                                          023A  1239 ;       :  Capture the Machine Check
                                          023A  1240 ;       :  IF the Machine Check is incorrect
                                          023A  1241 ;       :  :  THEN
                                          023A  1242 ;       :  :  Report error
                                          023A  1243 ;       :  ENDIF
                                          023A  1244 ;       ENDSUBTEST PCS Parity Error ABORT
                                          023A  1245 ;
                                          023A  1246 ;
                                          023A  1247 ;
                                          023A  1248 ;-
                                          023A  1249
                                          023A  1250
                                          023A  1251         $DS_BGNSUB
                                          023A       T14_S3::
     00010030 9F    0000004B'EF     FA    023A                       CALLG   $S$, @#DS$BGNSUB
                    00000014'EF     D4    0245  1252         CLRL    PCS$L_FLAGS               ; initialize flags
                    00000000'EF     D4    024B  1253         CLRL    PCS$L_COUNT               ; setup count for this subtest
                                          0251  1254         $DS_SETVEC_S    VECTOR=#SCB$L_MACHCHK,SRVADR=PCS$CHECK_MCHK
                             00     DD    0251                       PUSHL   #0
                    000004B8'EF     DF    0253                       PUSHAL  PCS$CHECK_MCHK
                             04     DD    0259                       PUSHL   #SCB$L_MACHCHK
           00010160 9F       03     FB    025B                       CALLS   #3, @#DS$SETVEC
           00000014'EF       20     88    0262  1255         BISB2   #PCS$M_EXP_CS_PAR,PCS$L_FLAGS
                                          0269  1256                                           ; Set flag specifying that we expect a
                                          0269  1257                                           ; control store parity error.
           56       0000000C'EF     DE    0269  1258         MOVAL   PCS$A_RETURN,R6           ; Load address of return address table
           66       0000029F'EF     DE    0270  1259         MOVAL   PCS$RENTRY4,(R6)          ; load rentry address CS Parity error
                                          0277  1260                                           ; machine check
        04 A6       000002BE'EF     DE    0277  1261         MOVAL   PCS$RENTRY5,4(R6)         ; Load rentry address for unexpected
                                          027F  1262                                           ; machine check (not CS parity error)
                                          027F  1263 PCS$TAG7:
           0000001C'FF       01     D0    027F  1264         MOVL    #1,@PCS$A_ENA_ADDRESS     ; Enable writing of patch bit
           57       00000020'EF     D0    0286  1265         MOVL    PCS$A_PTH_LOCATION,R7     ; load address of patch bit to enable
```

```
                          67   D4  028D  1266          CLRL     (R7)                           ; Clear (enable) the patch bit
               0000001C'FF     D4  028F  1267          CLRL     @PCS$A_ENA_ADDRESS             ; Disable writing of patch bit
               000003FC'EF     16  0295  1268          JSB      PCS$LOAD_MICRO                 ; load pcs with micro code
                                   029B  1269  PCS$PTH_INST:
                               01  029B  1270          NOP                                     ; The microword that perfroms the NOP
                                   029C  1271                                                  ; has the patch bit set with a next
                                   029C  1272                                                  ; address of 0002 which will cause a
                                   029C  1273                                                  ; trap to PCS 2802.
                        001F   31  029C  1274          BRW      PCS$RENTRY5                    ; If we end up here then there was no
                                   029F  1275                                                  ; parity error when executing the
                                   029F  1276                                                  ; patch Nop instruction
                                   029F  1277  PCS$RENTRY4:
     00000018'FF   000186A0 8F     CA  029F  1278          BICL2    #00100000,@PCS$A_PCS_ADDRESS
                                   02AA  1279                                                  ; disable PCS
                                   02AA  1280          $DS_CLRVEC_S     VECTOR=#SCB$L_MACHCHK
                          04   DD  02AA                         PUSHL    #SCB$L_MACHCHK
            00010168 9F   01   FB  02AC                         CALLS    #1, @#DS$CLRVEC
                                   02B3  1281          $DS_CLRVEC_S     VECTOR=#SCB$L_OPCCUS
                          14   DD  02B3                         PUSHL    #SCB$L_OPCCUS
            00010168 9F   01   FB  02B5                         CALLS    #1, @#DS$CLRVEC
                          46   11  02BC  1282          BRB      PCS$TAG8
                                   02BE  1283  PCS$RENTRY5:
     00000018'FF   000186A0 8F     CA  02BE  1284          BICL2    #00100000,@PCS$A_PCS_ADDRESS
                                   02C9  1285                                                  ; disable PCS
                                   02C9  1286          $DS_CLRVEC_S     VECTOR=#SCB$L_MACHCHK
                          04   DD  02C9                         PUSHL    #SCB$L_MACHCHK
            00010168 9F   01   FB  02CB                         CALLS    #1, @#DS$CLRVEC
                                   02D2  1287          $DS_CLRVEC_S     VECTOR=#SCB$L_OPCCUS
                          14   DD  02D2                         PUSHL    #SCB$L_OPCCUS
            00010168 9F   01   FB  02D4                         CALLS    #1, @#DS$CLRVEC
                                   02DB  1288          $DS_ERRHARD_S    UNIT=#0,PRLINK=PCS$NO_PARERR_MSG,P1=#PCS$T_PCS_ERR,-
                                   02DB  1289                           P2=#PCS$TABLE1
               00000210'8F     DD  02DB                         PUSHL    #PCS$TABLE1
               000005BB'8F     DD  02E1                         PUSHL    #PCS$T_PCS_ERR
               00000996'EF     DF  02E7                         PUSHAL   PCS$NO_PARERR_MSG
                          00   DD  02ED                         PUSHL    #0
                          00   DD  02EF                         PUSHL    #0
                          01   DD  02F1                         PUSHL    #$ER
                                   02F3        ;;; TEST 14, SUBTEST 3, ERROR 1
            000100D0 9F   06   FB  02F3                         CALLS    #$$M, @#DS$ERRHARD
                                   02FA  1290          $DS_BCOMPLETE    PCS$TAG8
                       07 50   E8  02FA                         BLBS     R0, PCS$TAG8
                                   02FD  1291          $DS_ABORT
            00010020 9F   6C   FA  02FD                         CALLG    (AP), @#DS$ABORT
                                   0304  1292  PCS$TAG8:
               000002CE'EF     16  0304  1293          JSB      PCS$CLEAR_PCS                  ; Initialize/Disable PCS
            0000FE08 9F   01   D1  030A  1294          CMPL     #1,@#DS$GL_PASSES              ; check to if more then one pass to run
                          07   13  0311  1295          BEQLU    PCS$NO_LOAD_PCS                ; br = only 1 pass specified, don't
                                   0313  1296                                                  ; load PCS
                                   0313  1297          $DS_LOADPCS_S                           ; Re-load patch micro code and patch
            000101C0 9F   00   FB  0313                         CALLS    #0, @#DS$LOADPCS
                                   031A  1298                                                  ; bits
                                   031A  1299  PCS$NO_LOAD_PCS:
                                   031A  1300          $DS_ENDSUB
                                   031A        T14_S3_X:
```

```
     00010038 9F    00000048'EF    FA  031A                         CALLG    $S$, @#DS$ENDSUB
                                       0325  1301
                                       0325  1302         $DS     TEST
                    50    01    D0     0325                         MOVL     #1, R0           ; NORMAL EXIT
                                       0328        TEST_014_X::
             00010058 9F    6E    FA   0328                         CALLG    (SP), @#DS$BREAK
                                 04    032F                         RET                       ; RETURN TO TEST SEQUENCER
                                       0330  1303
                                       0330  1304         $DS_ENDMOD
                                   00000000                         .PSECT   $TSTCNT, NOEXE, NOWRT, OVR, LONG
                          0000000E     0000                         .LONG    $TN
                                       0004  1305         .END
```

```
$$$                           = 00000048 R      05        DS$K_ERROR                  = 00000002
$$ARGS                        = 0000000A                  DS$K_NORMAL                 = 00000001
$$E                           = 00000003                  DS$K_SEVERE                 = 00000004
$$M                           = 00000006                  DS$K_SUBSYS                 = 00000066
$$N                           = 00000001                  DS$K_WARNING                = 00000000
$$S                           = FFFFFFFF                  DS$LOAD                       00010198
$$T1                          = 0000002C                  DS$LOADPCS                    000101C0
$ENV                          = 00000001  G               DS$MAPDBGBLOCK                00010118
$ER                           = FFFFFFFF                  DS$MEMSIZE                    000101D8
$MO                           = 00000001  G               DS$MMOFF                      00010158
$S$                           = 00000048 R      05        DS$MMON                       00010150
$ST                           = 00000000                  DS$MOVPHY                     00010148
$TN                           = 0000000E                  DS$MOVVRT                     00010140
ALL                           = 00000009  G               DS$PARSE                      000100B8
A_MCHK_BUFFER                   ********   X    01        DS$PRINTB                     000100E0
BASE_013                        00000000 R      03        DS$PRINTF                     000100F0
BASE_014                        00000000 R      06        DS$PRINTS                     000100F8
BIT...                        = 0000001A                  DS$PRINTSIG                   00010100
CEP_FUNCTIONAL                = 00000000                  DS$PRINTX                     000100E8
CEP_REPAIR                    = 00000001                  DS$PROBE                      000101A0
DATA_013                        00000014 R      03        DS$RELBUF                     00010128
DATA_014                        00000018 R      06        DS$RELMEM                     00010138
DEFAULT                       = 00000000  G               DS$SERVICE_DISPATCHER         000101D0
DS$ABORT                        00010020                  DS$SETIPL                     00010178
DS$ASKADR                       00010090                  DS$SETMAP                     00010188
DS$ASKDATA                      00010080                  DS$SETPRIEXV                  00010110
DS$ASKLGCL                      00010098                  DS$SETVEC                     00010160
DS$ASKSTR                       000100A0                  DS$SHOCHAN                    00010190
DS$ASKVLD                       00010088                  DS$SUMMARY                    00010028
DS$ATTACH                       000101A8                  DS$WAITMS                     00010060
DS$BGNSUB                       00010030                  DS$WAITUS                     00010068
DS$BRANCH                       000100A8                  DS$_ARITH                   = 006600D0
DS$BREAK                        00010058                  DS$_ASBE                    = 00660118
DS$CANWAIT                      00010070                  DS$_BADLINK                 = 006600F0
DS$CHANNEL                      00010180                  DS$_BADTYPE                 = 006600E8
DS$CKLOOP                       00010040                  DS$_BIIC                    = 00660120
DS$CLRVEC                       00010168                  DS$_CHME                    = 006600A8
DS$CNTRLC                       00010078                  DS$_CHMK                    = 006600E0
DS$CVTREG                       000100B0                  DS$_DEVNAME                 = 00660108
DS$ENDPASS                      00010010                  DS$_ERROR                   = 00660002
DS$ENDSUB                       00010038                  DS$_FHWE                    = 00660068
DS$ERRDEV                       000100C8                  DS$_FRAGBUF                 = 00660080
DS$ERRHARD                      000100D0                  DS$_ICBUSY                  = 006600C8
DS$ERRPREP                      00010108                  DS$_ICERR                   = 006600C0
DS$ERRSOFT                      000100D8                  DS$_IHWE                    = 00660060
DS$ERRSYS                       000100C0                  DS$_ILLCHAR                 = 00660018
DS$ESCAPE                       00010050                  DS$_ILLPAGCNT               = 00660078
DS$FREEDBGSYM                   000101B8                  DS$_ILLUNIT                 = 00660100
DS$GETBUF                       00010120                  DS$_INSFMEM                 = 00660050
DS$GETMEM                       00010130                  DS$_IPL2HI                  = 006600B8
DS$GETTERM                      000101C8                  DS$_IVADDR                  = 00660040
DS$GPHARD                       00010018                  DS$_IVVECT                  = 00660038
DS$HELP                         000101B0                  DS$_KRNLSTK                 = 00660090
DS$INITSCB                      00010170                  DS$_LOGIC                   = 00660070
DS$INLOOP                       00010048                  DS$_MCHK                    = 00660088
```

```
DS$_MMOFF                       = 00660058          ERR$_P3                         = 0000001C
DS$_NEEDUNIT                    = 006600F8          ERR$_P4                         = 00000020
DS$_NODE                        = 00660128          ERR$_P5                         = 00000024
DS$_NOPCS                       = 00660110          ERR$_P6                         = 00000028
DS$_NORMAL                      = 00660001          ERR$_POINTER                    = 00000010
DS$_NOSUPPORT                   = 006600B1          ERR$_UNIT                       = 00000008
DS$_NOTDON                      = 00660030          HALT                            = 00000004  G
DS$_NOTIMP                      = 006600B0          HP$A_DEPENDENT                    00000032
DS$_NULLSTR                     = 00660010          HP$A_DEVICE                       00000018
DS$_OVERFLOW                    = 00660008          HP$A_DVA                          0000001C
DS$_POWER                       = 00660098          HP$A_LINK                         00000020
DS$_PROGERR                     = 00660020          HP$B_DRIVE                        0000000B
DS$_SEVERE                      = 00660004          HP$B_FLAGS                        0000000A
DS$_TRANSL                      = 006600A0          HP$Q_DEVICE                       00000000
DS$_TRUNCATE                    = 00660028          HP$T_DEVICE                       0000000C
DS$_UNEXPINT                    = 006600D8          HP$T_TYPE                         00000026
DS$_VASFULL                     = 00660048          HP$W_SIZE                         00000008
DS$_WARNING                     = 00660000          HP$W_VECTOR                       00000024
DSA$AL_APTMAIL                    0000FE00          IPR                             = 00000006  G
DSA$AT_APTTXT                     0000FA00          KA$B_ACC_TYPE                     00000042  G
DSA$GL_APTCOM                     0000FE04          KA$B_NODE_ID                      00000046  G
DSA$GL_DEVLEN                     0000FE58          KA$B_RESERVED                     00000043  G
DSA$GL_ERRNO                      0000FE44          KA$B_SID_TYPE                     0000003D  G
DSA$GL_EVENT                      0000FE48          KA$K_LEN                          0000004B  G
DSA$GL_FLAGS                      0000FE00          KA$L_FLAGS                        00000032  G
DSA$GL_MSGTYP                     0000FE40          KA$L_MEM_SIZE                     00000047  G
DSA$GL_PASSES                     0000FE08          KA$L_SID                          0000003A  G
DSA$GL_PASSNO                     0000FE54          KA$M_CHAR                       = 00000100  G
DSA$GL_SECTNO                     0000FE10          KA$M_CRC                        = 00000010  G
DSA$GL_SID                        0000FE14          KA$M_D_FLOAT                    = 00000002  G
DSA$GL_SUBTNO                     0000FE4C          KA$M_EDIT                       = 00000080  G
DSA$GL_TESTNO                     0000FE50          KA$M_F_FLOAT                    = 00000001  G
DSA$GL_UNITS                      0000FE0C          KA$M_G_FLOAT                    = 00000004  G
DSA$GQ_MSGPTR                     0000FE68          KA$M_H_FLOAT                    = 00000008  G
DSA$GT_DEVNAM                     0000FE5C          KA$M_PACKED                     = 00000020  G
DSA$M_APT                       = 80000000          KA$M_PACK_MUL                   = 00000040  G
ENV$M_DOMAIN                    = 00000002          KA$M_SB_ERR                     = 02000000  G
ENV$M_LEVEL                     = 00000001          KA$M_TODR                       = 00010000  G
ENV$M_SUPER                     = 000003FC          KA$M_WCS_LD                     = 01000000  G
ENV$S_DOMAIN                    = 00000001          KA$V_CHAR                       = 00000008  G
ENV$S_LEVEL                     = 00000001          KA$V_CRC                        = 00000004  G
ENV$S_SUPER                     = 00000008          KA$V_D_FLOAT                    = 00000001  G
ENV$V_DOMAIN                    = 00000001          KA$V_EDIT                       = 00000007  G
ENV$V_LEVEL                     = 00000000          KA$V_F_FLOAT                    = 00000000  G
ENV$V_SUPER                     = 00000002          KA$V_G_FLOAT                    = 00000002  G
ENV$_CPU                        = 00000000          KA$V_H_FLOAT                    = 00000003  G
ENV$_FUNCTIONAL                 = 00000000          KA$V_PACKED                     = 00000005  G
ENV$_REPAIR                     = 00000001          KA$V_PACK_MUL                   = 00000006  G
ENV$_SUPER                      = 00000001          KA$V_SB_ERR                     = 00000019  G
ENV$_SYSTEM                     = 00000001          KA$V_TODR                       = 00000010  G
ERR$_MSGADR                     = 0000000C          KA$V_WCS_LD                     = 00000018  G
ERR$_NARGS                      = 0000000A          KA$W_LATENCY                      0000003E  G
ERR$_NUM                        = 00000004          KA$W_MEM_SIZE                     00000044  G
ERR$_P1                         = 00000014          KA$W_PROGRESS                     00000040  G
ERR$_P2                         = 00000018          KA$W_RESERVED                     00000038  G
```

```
KA$W_WCS                          00000036  G                 PCS$NOISY_WORDS                   00000130 R      01
L_REV_CHECK                       ********   X    03          PCS$NO_LOAD_PCS                   0000031A R      06
MANUAL                          = 00000001  G                 PCS$NO_PARERR_MSG                 00000996 R      01
MCHK                            = 00000005  G                 PCS$PAR_ERR_WORD                  000002B4 R      01
MSG_013                           00000002 R      03          PCS$PCS_ERR                       0000097B R      01
MSG_014                           00000002 R      06          PCS$PTH_INST                      0000029B R      06
PAR$M_ATDEF                     = 00000010                    PCS$RENTRY1                       000003EE R      03
PAR$M_ATHI                      = 00000008                    PCS$RENTRY2                       00000151 R      06
PAR$M_ATLO                      = 00000004                    PCS$RENTRY3                       000001FD R      06
PAR$M_NODEF                     = 00000002                    PCS$RENTRY4                       0000029F R      06
PAR$V_ATDEF                     = 00000004                    PCS$RENTRY5                       000002BE R      06
PAR$V_ATHI                      = 00000003                    PCS$RM_DECODE                     00000303 R      01
PAR$V_ATLO                      = 00000002                    PCS$ROWERR                        000008AF R      01
PAR$V_NODEF                     = 00000001                    PCS$SETUP_ADDR                    000003C4 R      01
PAR$_BIN                        = 00000002                    PCS$TABLE1                        00000210 R      01
PAR$_DEC                        = 0000000A                    PCS$TAG1                          000002E5 R      03
PAR$_HEX                        = 00000010                    PCS$TAG2                          0000040A R      03
PAR$_NO                         = 00000000                    PCS$TAG3                          0000010D R      06
PAR$_OCT                        = 00000008                    PCS$TAG4                          0000017D R      06
PAR$_YES                        = 00000001                    PCS$TAG5                          000001B5 R      06
PCS                             = 00000008  G                 PCS$TAG6                          00000229 R      06
PCS$ADDR_ERR                      000008FF R      01          PCS$TAG7                          0000027F R      06
PCS$A_ENA_ADDRESS                 0000001C R      01          PCS$TAG8                          00000304 R      06
PCS$A_PCS_ADDRESS                 00000018 R      01          PCS$T_ADR_ERR                     0000076C R      01
PCS$A_PCS_BUF                     00000030 R      01          PCS$T_CS_PARITY1                  000004FF R      01
PCS$A_PCS_BUFFER                  ********   X    01          PCS$T_CS_PARITY2                  00000532 R      01
PCS$A_PTHBIT_ADDRESS              00000024 R      01          PCS$T_CS_PARITY3                  0000055E R      01
PCS$A_PTH_LOCATION                00000020 R      01          PCS$T_DATAPATH                    0000068B R      01
PCS$A_RETURN                      0000000C R      01          PCS$T_DPERR                       000006B1 R      01
PCS$CHECK_MCHK                    000004B8 R      01          PCS$T_ENTRY_ERR                   00000805 R      01
PCS$CLEAR_BUFFER                  000002B8 R      01          PCS$T_EXPECTED                    00000616 R      01
PCS$CLEAR_PCS                     000002CE R      01          PCS$T_MC_ERR                      000005E5 R      01
PCS$CODEVECTORS                   000001B0 R      01          PCS$T_MC_MESS                     00000592 R      01
PCS$CODE_WORDS                    000001C0 R      01          PCS$T_MI_ERR1                     0000079A R      01
PCS$DATA_PATTERN                  00000028 R      01          PCS$T_MI_ERR2                     000007D0 R      01
PCS$DATA_TABLE                    000001EC R      01          PCS$T_NO_PCS                      00000623 R      01
PCS$DECODED_WORDS                 000000B0 R      01          PCS$T_NXFIELD_ERR                 0000082A R      01
PCS$DPERR                         0000088C R      01          PCS$T_PCS_ERR                     000005BB R      01
PCS$F_PCS_ERR                     00000858 R      01          PCS$T_PCS_NORESPOND               0000064D R      01
PCS$F_ROWERR1                     0000073C R      01          PCS$T_ROW                         000006EA R      01
PCS$F_ROWERR2                     00000757 R      01          PCS$T_ROWERR                      00000711 R      01
PCS$LOAD_MICRO                    000003FC R      01          PCS$V_BIT14                     = 0000000E
PCS$L_COUNT                       00000000 R      01          PCS$V_DATA_ERR                  = 00000001
PCS$L_FLAGS                       00000014 R      01          PCS$V_EXP_CS_PAR                = 00000005
PCS$L_INC                         00000004 R      01          PCS$V_PAT                       = 00000000
PCS$L_ROW                         00000008 R      01          PCS$V_REVERSE                   = 00000003
PCS$MICRO_DATA                    0000024C R      01          PCS$V_ROW_ERR                   = 00000002
PCS$MI_ERR                        00000955 R      01          POWER_FAIL                      = 00000003  G
PCS$M_BIT19                     = 00080000                    PR$_SID                           ********   X    03
PCS$M_BIT20                     - 00100000                    PR750$_CADR                     = 00000025
PCS$M_DATA_ERR                  = 00000002                    PSL$K_EXEC                      = 00000001
PCS$M_EXP_CS_PAR                = 00000020                    PSL$K_KERNEL                    = 00000000
PCS$M_PAT                       = 00000001                    PSL$K_SUPER                     = 00000002
PCS$M_REVERSE                   = 00000008                    PSL$K_USER                      = 00000003
PCS$M_ROW_ERR                   = 00000004                    PSL$M_CBIT                      = 00000001
```

```
PSL$M_NBIT                       = 00000008        SYS$CONNECT                      000105C0
PSL$M_VBIT                       = 00000002        SYS$DALLOC                       000102D8
PSL$M_ZBIT                       = 00000004        SYS$DASSGN                       000102E0
PSL$V_CBIT                       = 00000000        SYS$DISCONNECT                   000105D0
PSL$V_NBIT                       = 00000003        SYS$FAO                          00010350
PSL$V_VBIT                       = 00000001        SYS$FAOL                         00010358
PSL$V_ZBIT                       = 00000002        SYS$GET                          00010580
SCB$L_ACCESS                       00000020        SYS$GETCHN                       000104C8
SCB$L_ARITH                        00000034        SYS$GETTIM                       00010378
SCB$L_BREAK                        0000002C        SYS$LKWSET                       000103A0
SCB$L_CHME                         00000044        SYS$NUMTIM                       000103B8
SCB$L_CHMK                         00000040        SYS$OPEN                         00010608
SCB$L_CHMS                         00000048        SYS$QIO                          000103C8
SCB$L_CHMU                         0000004C        SYS$QIOW                         00010200
SCB$L_COMPAT                       00000030        SYS$READ                         00010590
SCB$L_KNLSTK                       00000008        SYS$READEF                       000103D0
SCB$L_MACHCHK                      00000004        SYS$SETAST                       000103F8
SCB$L_OPCCUS                       00000014        SYS$SETEF                        00010400
SCB$L_OPCDEC                       00000010        SYS$SETIMR                       00010420
SCB$L_POWER                        0000000C        SYS$SETPRI                       00010428
SCB$L_RADRMOD                      0000001C        SYS$SETPRT                       00010430
SCB$L_ROPRAND                      00000018        SYS$SETRWM                       00010438
SCB$L_RXDB                         000000F8        SYS$SETSWM                       00010448
SCB$L_SFTLVL1                      00000084        SYS$ULKPAG                       00010460
SCB$L_SFTLVL10                     000000A8        SYS$ULWSET                       00010468
SCB$L_SFTLVL11                     000000AC        SYS$UNWIND                       00010470
SCB$L_SFTLVL12                     000000B0        SYS$WAITFR                       00010478
SCB$L_SFTLVL13                     000000B4        SYS$WFLAND                       00010488
SCB$L_SFTLVL14                     000000B8        SYS$WFLOR                        00010490
SCB$L_SFTLVL15                     000000BC        T13_S1                           000000DF RG     03
SCB$L_SFTLVL2                      00000088        T13_S1_X                         00000177 R      03
SCB$L_SFTLVL3                      0000008C        T13_S2                           00000182 RG     03
SCB$L_SFTLVL4                      00000090        T13_S2_X                         00000215 R      03
SCB$L_SFTLVL5                      00000094        T13_S3                           00000220 RG     03
SCB$L_SFTLVL6                      00000098        T13_S3_X                         000002B3 R      03
SCB$L_SFTLVL7                      0000009C        T13_S4                           000002BE RG     03
SCB$L_SFTLVL8                      000000A0        T13_S4_X                         00000417 R      03
SCB$L_SFTLVL9                      000000A4        T14_S1                           000000EB RG     06
SCB$L_TBIT                         00000028        T14_S1_X                         00000183 R      06
SCB$L_TIMER                        000000C0        T14_S2                           0000018E RG     06
SCB$L_TRANSL                       00000024        T14_S2_X                         0000022F R      06
SCB$L_TXDB                         000000FC        T14_S3                           0000023A RG     06
SCB$L_WTBUSERR                   = 00000060        T14_S3_X                         0000031A R      06
SCB$L_ZERO                         00000000        TEST_013                         00000018 RG     03
SEP_FUNCTIONAL                   = 00000002        TEST_013_X                       00000425 RG     03
SEP_REPAIR                       = 00000003        TEST_014                         0000001C RG     06
SIZ...                           = 00000001        TEST_014_X                       00000328 RG     06
SYS$ALLOC                          00010238        TU58                           = 00000002  G
SYS$ASCTIM                         00010248        WCS                            = 00000007  G
SYS$ASSIGN                         00010250
SYS$BINTIM                         00010258
SYS$CANCEL                         00010260
SYS$CANTIM                         00010268
SYS$CLOSE                          000105B8
SYS$CLREF                          00010298
```

```
                                                +-----------------+
                                                ! Psect synopsis !
                                                +-----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
ECKAXH                          000009DF  ( 2527.)  01 (  1.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD  NOWRT NOVEC PAGE
$ABS$                           00010610  (67088.)  02 (  2.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
TEST_013                        0000042D  ( 1069.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD  NOWRT NOVEC PAGE
DISPATCH                        00000030  (   48.)  04 (  4.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD  NOWRT NOVEC LONG
ARGLIST                         00000054  (   84.)  05 (  5.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD  NOWRT NOVEC LONG
TEST_014                        00000330  (  816.)  06 (  6.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD  NOWRT NOVEC PAGE
$TSTCNT                         00000004  (    4.)  07 (  7.)  NOPIC   USR   OVR   REL   LCL NOSHR NOEXE   RD  NOWRT NOVEC LONG
```

```
                                        +-------------------------+
                                        ! Symbol Cross Reference !
                                        +-------------------------+

SYMBOL                         VALUE           DEFINITION          REFERENCES...
------                         -----           ----------          -------------
$$$                            =00000048-R     1251    (16)        1125    (14)        1188    (15)        1251    (16)        742     (9)
                                                                   815     (10)        886     (11)        982     (12)
$$ARGS                         =0000000A       83      (2)         83      (2)
$$E                            =00000003       1251    (16)        1046    (12)        1152    (14)        1215    (15)        1300    (16)
                                                                   771     (9)         844     (10)        915     (11)
$$M                            =00000006       1289    (16)        1004    (12)        1011    (12)        1042    (12)        1061    (13)
                                                                   1071    (13)        1083    (13)        1143    (14)        1147    (14)
                                                                   1206    (15)        1210    (15)        1289    (16)        657     (8)
                                                                   667     (8)         679     (8)         757     (9)         841     (10)
                                                                   910     (11)
$$N                            =00000001       1289    (16)        1004    (12)        1011    (12)        1042    (12)        1071    (13)
                                                                 #-1074    (13)        1083    (13)      #-1087    (13)        1143    (14)
                                                                   1147    (14)        1206    (15)        1210    (15)        1289    (16)
                                                                   577     (7)       #-582     (7)         588     (7)         590     (7)
                                                                 #-598     (7)         606     (7)         608     (7)         620     (7)
                                                                   627     (7)       #-632     (7)       #-633     (7)       #-634     (7)
                                                                 #-638     (7)         667     (8)       #-670     (8)         679     (8)
                                                                 #-683     (8)         757     (9)         841     (10)        87      (2)
                                                                   910     (11)
$$S                            =FFFFFFFF       1302    (16)        1052    (13)        1061    (13)        1096    (14)        1125    (14)
                                                                   1158    (15)        1188    (15)        1218    (16)        1251    (16)
                                                                   645     (8)         657     (8)         692     (9)         742     (9)
                                                                   775     (10)        815     (10)        848     (11)        886     (11)
                                                                   919     (12)        982     (12)
$$T1                           =0000002C       83      (2)         83      (2)
$ENV                           =00000001       66      (2)
$ER                            =FFFFFFFF       1302    (16)      #-1004    (12)      #-1011    (12)      #-1042    (12)      #-1071    (13)
                                                                 #-1083    (13)        1125    (14)      #-1143    (14)      #-1147    (14)
                                                                   1188    (15)      #-1206    (15)      #-1210    (15)        1251    (16)
                                                                 #-1289    (16)      #-667     (8)       #-679     (8)         742     (9)
                                                                 #-757     (9)         815     (10)      #-841     (10)        886     (11)
                                                                 #-910     (11)        982     (12)
$MO                            =00000001       1304    (16)        1304    (16)
$S$                            =00000048-R     1251    (16)        1046    (12)        1125    (14)        1152    (14)        1188    (15)
                                                                   1215    (15)        1251    (16)        1300    (16)        742     (9)
                                                                   771     (9)         815     (10)        844     (10)        886     (11)
                                                                   915     (11)        982     (12)
$ST                            =00000000       1302    (16)        1004    (12)        1011    (12)        1042    (12)        1043    (12)
                                                                   1046    (12)        1048    (12)        1061    (13)        1071    (13)
                                                                   1083    (13)        1096    (14)        1125    (14)        1143    (14)
                                                                   1147    (14)        1149    (14)        1152    (14)        1158    (15)
                                                                   1188    (15)        1206    (15)        1210    (15)        1212    (15)
                                                                   1215    (15)        1218    (16)        1251    (16)        1289    (16)
                                                                   1300    (16)        1302    (16)        657     (8)         667     (8)
                                                                   679     (8)         687     (8)         692     (9)         742     (9)
                                                                   757     (9)         758     (9)         771     (9)         775     (10)
                                                                   815     (10)        841     (10)        844     (10)        848     (11)
                                                                   886     (11)        910     (11)        915     (11)        919     (12)
```

```
                                                             982    (12)
$TN                            =0000000E     1304   (16)     1004   (12)     1011   (12)     1042   (12)     1048   (12)
                                                             1052   (13)     1061   (13)     1071   (13)     1073   (13)
                                                             1075   (13)     1083   (13)     1086   (13)     1088   (13)
                                                             1096   (14)     1143   (14)     1147   (14)     1158   (15)
                                                             1206   (15)     1210   (15)     1218   (16)     1289   (16)
                                                             1302   (16)     1304   (16)     645    (8)      657    (8)
                                                             667    (8)      669    (8)      671    (8)      679    (8)
                                                             682    (8)      684    (8)      692    (9)      757    (9)
                                                             775    (10)     841    (10)     848    (11)     910    (11)
                                                             919    (12)
ALL                            =00000009     87     (2)      1061   (13)     657    (8)
A_MCHK_BUFFER                   00000000-XR                #-487    (3)    #-635    (7)
BASE_013                        00000000-R   645    (8)      657    (8)
BASE_014                        00000000-R   1052   (13)     1061   (13)
BIT...                         =0000001A     82     (2)      66     (2)      80     (2)      81     (2)      82     (2)
CEP_FUNCTIONAL                 =00000000     66     (2)
CEP_REPAIR                     =00000001     66     (2)      66     (2)
DATA_013                        00000014-R   657    (8)      657    (8)
DATA_014                        00000018-R   1061   (13)     1061   (13)
DEFAULT                        =00000000     87     (2)      1061   (13)     657    (8)
DS$ABORT                        00010020                     1291   (16)     476    (3)
DS$BGNSUB                       00010030                     1125   (14)     1188   (15)     1251   (16)     742    (9)
                                                             815    (10)     886    (11)     982    (12)
DS$BREAK                        00010058                     1012   (12)     1048   (12)     1302   (16)
DS$CKLOOP                       00010040                     1043   (12)     1149   (14)     1212   (15)     687    (8)
                                                             758    (9)
DS$CLRVEC                       00010168                     1045   (12)     1280   (16)     1281   (16)     1286   (16)
                                                             1287   (16)
DS$ENDSUB                       00010038                     1046   (12)     1152   (14)     1215   (15)     1300   (16)
                                                             771    (9)      844    (10)     915    (11)
DS$ERRHARD                      000100D0                     1004   (12)     1011   (12)     1042   (12)     1071   (13)
                                                             1083   (13)     1143   (14)     1147   (14)     1206   (15)
                                                             1210   (15)     1289   (16)     667    (8)      679    (8)
                                                             757    (9)      841    (10)     910    (11)
DS$K_ERROR                     =00000002     80     (2)
DS$K_NORMAL                    =00000001     80     (2)
DS$K_SEVERE                    =00000004     80     (2)
DS$K_SUBSYS                    =00000066     80     (2)      80     (2)
DS$K_WARNING                   =00000000     80     (2)
DS$LOADPCS                      000101C0                     1297   (16)
DS$PRINTB                       000100E0                     577    (7)      582    (7)      588    (7)      598    (7)
                                                             606    (7)      620    (7)      627    (7)      632    (7)
                                                             633    (7)      634    (7)      638    (7)
DS$PRINTF                       000100F0                     1074   (13)     1087   (13)     670    (8)      683    (8)
DS$PRINTSIG                     00010100                     475    (3)      637    (7)      639    (7)
DS$PRINTX                       000100E8                     590    (7)      608    (7)
DS$PROBE                        000101A0                     1077   (13)     673    (8)
DS$SETVEC                       00010160                     1090   (13)     1254   (16)     984    (12)
DS$_ARITH                      =006600D0     80     (2)
DS$_ASBE                       =00660118     80     (2)
DS$_BADLINK                    =006600F0     80     (2)
DS$_BADTYPE                    =006600E8     80     (2)
DS$_BIIC                       =00660120     80     (2)
DS$_CHME                       =006600A8     80     (2)
```

```
DS$_CHMK                       =006600E0     80     (2)
DS$_DEVNAME                    =00660108     80     (2)
DS$_ERROR                      =00660002     80     (2)
DS$_FHWE                       =00660068     80     (2)
DS$_FRAGBUF                    =00660080     80     (2)
DS$_ICBUSY                     =006600C8     80     (2)
DS$_ICERR                      =006600C0     80     (2)
DS$_IHWE                       =00660060     80     (2)
DS$_ILLCHAR                    =00660018     80     (2)
DS$_ILLPAGCNT                  =00660078     80     (2)
DS$_ILLUNIT                    =00660100     80     (2)
DS$_INSFMEM                    =00660050     80     (2)
DS$_IPL2HI                     =006600B8     80     (2)
DS$_IVADDR                     =00660040     80     (2)
DS$_IVVECT                     =00660038     80     (2)
DS$_KRNLSTK                    =00660090     80     (2)
DS$_LOGIC                      =00660070     80     (2)
DS$_MCHK                       =00660088     80     (2)      172    (2)    #-474    (3)
DS$_MMOFF                      =00660058     80     (2)
DS$_NEEDUNIT                   =006600F8     80     (2)
DS$_NODE                       =00660128     80     (2)
DS$_NOPCS                      =00660110     80     (2)
DS$_NORMAL                     =00660001     80     (2)
DS$_NOSUPPORT                  =006600B1     80     (2)
DS$_NOTDON                     =00660030     80     (2)
DS$_NOTIMP                     =006600B0     80     (2)      80     (2)
DS$_NULLSTR                    =00660010     80     (2)
DS$_OVERFLOW                   =00660008     80     (2)
DS$_POWER                      =00660098     80     (2)
DS$_PROGERR                    =00660020     80     (2)
DS$_SEVERE                     =00660004     80     (2)
DS$_TRANSL                     =006600A0     80     (2)
DS$_TRUNCATE                   =00660028     80     (2)
DS$_UNEXPINT                   =006600D8     80     (2)
DS$_VASFULL                    =00660048     80     (2)
DS$_WARNING                    =00660000     80     (2)      80     (2)
DSA$GL_FLAGS                    0000FE00                   #-1069   (13)   #-1081   (13)   #-665    (8)    #-677    (8)
DSA$GL_PASSES                   0000FE08                   #-1294   (16)
DSA$GL_PASSNO                   0000FE54                   #-1066   (13)   #-1079   (13)   #-662    (8)    #-675    (8)
DSA$M_APT                      =80000000                   #-1069   (13)   #-1081   (13)   #-665    (8)    #-677    (8)
ENV$M_DOMAIN                   =00000002     66     (2)
ENV$M_LEVEL                    =00000001     66     (2)
ENV$M_SUPER                    =000003FC     66     (2)
ENV$S_DOMAIN                   =00000001     66     (2)
ENV$S_LEVEL                    =00000001     66     (2)
ENV$S_SUPER                    =00000008     66     (2)
ENV$V_DOMAIN                   =00000001     66     (2)      66     (2)
ENV$V_LEVEL                    =00000000     66     (2)      66     (2)
ENV$V_SUPER                    =00000002     66     (2)
ENV$_CPU                       =00000000     66     (2)      66     (2)
ENV$_FUNCTIONAL                =00000000     66     (2)      66     (2)
ENV$_REPAIR                    =00000001     66     (2)      66     (2)
ENV$_SUPER                     =00000001     66     (2)
ENV$_SYSTEM                    =00000001     66     (2)      66     (2)
ERR$_MSGADR                    =0000000C     83     (2)
```

```
ERR$_NARGS                      =0000000A    83     (2)
ERR$_NUM                        =00000004    83     (2)
ERR$_P1                         =00000014    83     (2)     #-577   (7)     #-600   (7)     #-616   (7)       633   (7)
ERR$_P2                         =00000018    83     (2)       574   (7)     #-617   (7)       639   (7)
ERR$_P3                         =0000001C    83     (2)     #-575   (7)     #-577   (7)     #-618   (7)
ERR$_P4                         =00000020    83     (2)
ERR$_P5                         =00000024    83     (2)
ERR$_P6                         =00000028    83     (2)
ERR$_POINTER                    =00000010    83     (2)
ERR$_UNIT                       =00000008    83     (2)
HALT                            =00000004    87     (2)
IPR                             =00000006    87     (2)
L_REV_CHECK                      00000000-XR                #-1064   (13)    #-660   (8)
MANUAL                          =00000001    87     (2)
MCHK                            =00000005    87     (2)
MSG_013                          00000002-R  645    (8)       657   (8)
MSG_014                          00000002-R  1052   (13)     1061   (13)
PAR$M_ATDEF                     =00000010    81     (2)
PAR$M_ATHI                      =00000008    81     (2)
PAR$M_ATLO                      =00000004    81     (2)
PAR$M_NODEF                     =00000002    81     (2)
PAR$V_ATDEF                     =00000004    81     (2)
PAR$V_ATHI                      =00000003    81     (2)
PAR$V_ATLO                      =00000002    81     (2)
PAR$V_NODEF                     =00000001    81     (2)
PAR$_BIN                        =00000002    81     (2)
PAR$_DEC                        =0000000A    81     (2)
PAR$_HEX                        =00000010    81     (2)
PAR$_NO                         =00000000    81     (2)
PAR$_OCT                        =00000008    81     (2)
PAR$_YES                        =00000001    81     (2)
PCS                             =00000008    87     (2)      1061   (13)      657   (8)
PCS$ADDR_ERR                     000008FF-R  596    (7)       910   (11)
PCS$A_ENA_ADDRESS                0000001C-R  120    (2)    #-1131   (14)   #-1134   (14)   #-1194   (15)   #-1197   (15)
                                                           #-1264   (16)   #-1267   (16)    #-283   (3)     #-288   (3)
                                                            #-393   (3)     #-745   (9)     #-889   (11)
PCS$A_PCS_ADDRESS                00000018-R  117    (2)    #-1076   (13)   #-1140   (14)   #-1146   (14)   #-1203   (15)
                                                           #-1209   (15)   #-1278   (16)   #-1284   (16)    #-287   (3)
                                                            #-367   (3)     #-395   (3)     #-446   (3)     #-449   (3)
                                                            #-451   (3)     #-584   (7)     #-672   (8)     #-744   (9)
                                                            #-818   (10)    #-819   (10)    #-824   (10)    #-890   (11)
PCS$A_PCS_BUF                    00000030-R  133    (2)       271   (3)       585   (7)       826   (10)      833   (10)
PCS$A_PCS_BUFFER                 00000000-XR                #-365   (3)     #-997   (12)
PCS$A_PTHBIT_ADDRESS             00000024-R  126    (2)     #-282   (3)
PCS$A_PTH_LOCATION               00000020-R  123    (2)    #-1132   (14)   #-1195   (15)   #-1265   (16)
PCS$A_RETURN                     0000000C-R  96     (2)    #-1128   (14)   #-1191   (15)     1258   (16)    #-481   (3)
                                                            #-493   (3)     #-983   (12)
PCS$CHECK_MCHK                   000004B8-R  467    (3)      1090   (13)     1254   (16)      984   (12)
PCS$CLEAR_BUFFER                 000002B8-R  270    (3)       817   (10)
PCS$CLEAR_PCS                    000002CE-R  281    (3)      1091   (13)     1148   (14)     1151   (14)     1211   (15)
                                                             1214   (15)     1293   (16)      689   (8)       770   (9)
                                                              843   (10)      913   (11)
PCS$CODEVECTORS                  000001B0-R  141    (2)     #-342   (3)
PCS$CODE_WORDS                   000001C0-R  147    (2)       893   (11)
PCS$DATA_PATTERN                 00000028-R  129    (2)     #-747   (9)     #-764   (9)
```

```
PCS$DATA_TABLE                000001EC-R    159     (2)       990    (12)       991    (12)
PCS$DECODED_WORDS             000000B0-R    135     (2)     #-302    (3)        332    (3)      #-340     (3)        352    (3)
                                                              602    (7)        902    (11)
PCS$DPERR                     0000088C-R    572     (7)       757    (9)
PCS$F_PCS_ERR                 00000858-R    562     (5)       627    (7)
PCS$F_ROWERR1                 0000073C-R    541     (5)       588    (7)        606    (7)
PCS$F_ROWERR2                 00000757-R    544     (5)       590    (7)        608    (7)
PCS$LOAD_MICRO                000003FC-R    392     (3)      1135    (14)      1198    (15)      1268     (16)
PCS$L_COUNT                   00000000-R    93      (2)    #-1127    (14)    #-1190    (15)    #-1253     (16)     #-272    (3)
                                                            #-274    (3)      #-284    (3)      #-286     (3)      #-289    (3)
                                                            #-291    (3)      #-402    (3)      #-439     (3)
PCS$L_FLAGS                   00000014-R    98      (2)      1013    (12)      1026    (12)    #-1028     (12)    #-1126    (14)
                                                           #-1189    (15)    #-1252    (16)    #-1255     (16)       443    (3)
                                                              470    (3)      #-743    (9)        762     (9)      #-765    (9)
                                                            #-768    (9)      #-816    (10)     #-836     (10)       839    (10)
                                                            #-887    (11)     #-905    (11)       908     (11)     #-986    (12)
                                                              992    (12)
PCS$L_INC                     00000004-R    94      (2)    #-1031    (12)      1032    (12)     #-364     (3)      #-988    (12)
PCS$L_ROW                     00000008-R    95      (2)    #-1035    (12)    #-1037    (12)     #-368     (3)      #-987    (12)
PCS$MICRO_DATA                0000024C-R    202     (2)       396    (3)
PCS$MI_ERR                    00000955-R    614     (7)      1004    (12)      1011    (12)
PCS$M_BIT19                  =00080000      114     (2)     #-427    (3)
PCS$M_BIT20                  =00100000      115     (2)     #-446    (3)      #-451    (3)
PCS$M_DATA_ERR               =00000002      103     (2)     #-905    (11)
PCS$M_EXP_CS_PAR             =00000020      109     (2)    #-1255    (16)
PCS$M_PAT                    =00000001      101     (2)     #-765    (9)      #-768    (9)
PCS$M_REVERSE                =00000008      107     (2)    #-1028    (12)
PCS$M_ROW_ERR                =00000004      105     (2)     #-836    (10)
PCS$NOISY_WORDS               00000130-R    138     (2)     #-310    (3)      #-338    (3)        894     (11)
PCS$NO_LOAD_PCS               0000031A-R    1299    (16)   #-1295    (16)
PCS$NO_PARERR_MSG             00000996-R    630     (7)      1289    (16)
PCS$PAR_ERR_WORD              000002B4-R    250     (2)     #-450    (3)
PCS$PCS_ERR                   0000097B-R    623     (7)      1143    (14)      1206    (15)
PCS$PTH_INST                  0000029B-R    1269    (16)      176    (2)        184    (2)
PCS$RENTRY1                   000003EE-R    1041    (12)      983    (12)
PCS$RENTRY2                   00000151-R    1145    (14)     1128    (14)
PCS$RENTRY3                   000001FD-R    1208    (15)     1191    (15)
PCS$RENTRY4                   0000029F-R    1277    (16)     1259    (16)
PCS$RENTRY5                   000002BE-R    1283    (16)     1261    (16)    #-1274    (16)
PCS$RM_DECODE                 00000303-R    298     (3)       900    (11)
PCS$ROWERR                    000008AF-R    580     (7)       841    (10)
PCS$SETUP_ADDR                000003C4-R    363     (3)       989    (12)
PCS$TABLE1                    00000210-R    170     (2)    #-1289    (16)     #-636    (7)
PCS$TAG1                      000002E5-R    985     (12)     1043    (12)
PCS$TAG2                      0000040A-R    1044    (12)   #-1040    (12)
PCS$TAG3                      0000010D-R    1129    (14)     1149    (14)
PCS$TAG4                      0000017D-R    1150    (14)   #-1142    (14)    #-1144    (14)
PCS$TAG5                      000001B5-R    1192    (15)     1212    (15)
PCS$TAG6                      00000229-R    1213    (15)   #-1205    (15)    #-1207    (15)
PCS$TAG7                      0000027F-R    1263    (16)
PCS$TAG8                      00000304-R    1292    (16)   #-1282    (16)    #-1290    (16)
PCS$T_ADR_ERR                 0000076C-R    547     (5)       910    (11)
PCS$T_CS_PARITY1              000004FF-R    501     (5)      1042    (12)
PCS$T_CS_PARITY2              00000532-R    504     (5)      1147    (14)
PCS$T_CS_PARITY3              0000055E-R    507     (5)      1210    (15)
```

```
PCS$T_DATAPATH                  0000068B-R    529     (5)       757     (9)
PCS$T_DPERR                     000006B1-R    532     (5)       577     (7)       620     (7)
PCS$T_ENTRY_ERR                 00000805-R    556     (5)       1143    (14)
PCS$T_EXPECTED                  00000616-R    520     (5)       638     (7)
PCS$T_MC_ERR                    000005E5-R    516     (5)       634     (7)
PCS$T_MC_MESS                   00000592-R    510     (5)       632     (7)
PCS$T_MI_ERR1                   0000079A-R    550     (5)       1004    (12)
PCS$T_MI_ERR2                   000007D0-R    553     (5)       1011    (12)
PCS$T_NO_PCS                    00000623-R    523     (5)       1071    (13)      1074    (13)      667     (8)       670     (8)
PCS$T_NXFIELD_ERR               0000082A-R    559     (5)       1206    (15)
PCS$T_PCS_ERR                   000005BB-R    513     (5)      #-1289   (16)
PCS$T_PCS_NORESPOND             0000064D-R    526     (5)       1083    (13)      1087    (13)      679     (8)       683     (8)
PCS$T_ROW                       000006EA-R    535     (5)       841     (10)
PCS$T_ROWERR                    00000711-R    538     (5)       582     (7)       598     (7)
PCS$V_BIT14                    =0000000E      113     (2)      #-1032   (12)     #-371    (3)      #-899    (11)
PCS$V_DATA_ERR                 =00000001      102     (2)      #-908    (11)
PCS$V_EXP_CS_PAR               =00000005      108     (2)      #-443    (3)      #-470    (3)
PCS$V_PAT                      =00000000      100     (2)      #-762    (9)
PCS$V_REVERSE                  =00000003      106     (2)      #-1013   (12)     #-1026   (12)     #-992    (12)
PCS$V_ROW_ERR                  =00000002      104     (2)      #-839    (10)
POWER_FAIL                     =00000003      87      (2)
PR$_SID                         00000000-XR                    #-1062   (13)     #-658    (8)
PR750$_CADR                    =00000025                       #-1089   (13)     #-686    (8)
PSL$K_EXEC                     =00000001      82      (2)
PSL$K_KERNEL                   =00000000      82      (2)
PSL$K_SUPER                    =00000002      82      (2)
PSL$K_USER                     =00000003      82      (2)
PSL$M_CBIT                     =00000001      82      (2)
PSL$M_NBIT                     =00000008      82      (2)
PSL$M_VBIT                     =00000002      82      (2)
PSL$M_ZBIT                     =00000004      82      (2)
PSL$V_CBIT                     =00000000      82      (2)
PSL$V_NBIT                     =00000003      82      (2)
PSL$V_VBIT                     =00000001      82      (2)
PSL$V_ZBIT                     =00000002      82      (2)
SCB$L_MACHCHK                   00000004                       #-1045   (12)     #-1090   (13)     #-1254   (16)     #-1280   (16)
                                                               #-1286   (16)     #-984    (12)
SCB$L_OPCCUS                    00000014                       #-1281   (16)     #-1287   (16)
SCB$L_WTBUSERR                 =00000060      112     (2)
SEP_FUNCTIONAL                 =00000002      66      (2)
SEP_REPAIR                     =00000003      66      (2)
SIZ...                         =00000001      82      (2)       66      (2)       81      (2)       82      (2)
T13_S1                          000000DF-R    742     (9)
T13_S1_X                        00000177-R    771     (9)
T13_S2                          00000182 R    815     (10)
T13_S2_X                        00000215-R    844     (10)
T13_S3                          00000220-R    886     (11)
T13_S3_X                        000002B3-R    915     (11)
T13_S4                          000002BE-R    982     (12)
T13_S4_X                        00000417-R    1046    (12)
T14_S1                          000000EB-R    1125    (14)
T14_S1_X                        00000183-R    1152    (14)
T14_S2                          0000018E-R    1188    (15)
T14_S2_X                        0000022F-R    1215    (15)
T14_S3                          0000023A-R    1251    (16)
```

```
T14_S3_X                         0000031A-R    1300    (16)
TEST_013                         00000018-R    657     (8)       657     (8)
TEST_013_X                       00000425-R    1048    (12)    #-669     (8)      #-671     (8)      #-682     (8)      #-684     (8)
TEST_014                         0000001C-R    1061    (13)      1061    (13)
TEST_014_X                       00000328-R    1302    (16)    #-1073    (13)     #-1075    (13)     #-1086    (13)     #-1088    (13)
TU58                            =00000002      87      (2)
WCS                             =00000007      87      (2)
```

```
                                             +-------------------------+
                                             ! Macros Cross Reference !
                                             +-------------------------+

MACRO            SIZE          DEFINITION     REFERENCES...
-----            ----          ----------     -------------
$D1_ABPROGRAM    1             476    (3)     1291   (16)     476    (3)
$D1_BSUBT        1             742    (9)     1125   (14)     1188   (15)     1251   (16)     742    (9)      815    (10)
                                              886    (11)     982    (12)
$D1_BTEST        1             657    (8)     1061   (13)     657    (8)
$D1_ERR          1             667    (8)     1004   (12)     1011   (12)     1042   (12)     1071   (13)     1083   (13)
                                              1143   (14)     1147   (14)     1206   (15)     1210   (15)     1289   (16)
                                              667    (8)      679    (8)      757    (9)      841    (10)     910    (11)
$D1_ERR_S        1             667    (8)     1004   (12)     1011   (12)     1042   (12)     1071   (13)     1083   (13)
                                              1143   (14)     1147   (14)     1206   (15)     1210   (15)     1289   (16)
                                              667    (8)      679    (8)      757    (9)      841    (10)     910    (11)
$D1_ESUBT        1             771    (9)     1046   (12)     1152   (14)     1215   (15)     1300   (16)     771    (9)
                                              844    (10)     915    (11)
$D1_ETEST        1             1048   (12)    1048   (12)     1302   (16)
$D1_EXTEST       1             669    (8)     1073   (13)     1075   (13)     1086   (13)     1088   (13)     669    (8)
                                              671    (8)      682    (8)      684    (8)
$D1_PRINT_S      1             577    (7)     1074   (13)     1087   (13)     577    (7)      582    (7)      588    (7)
                                              590    (7)      598    (7)      606    (7)      608    (7)      620    (7)
                                              627    (7)      632    (7)      633    (7)      634    (7)      638    (7)
                                              670    (8)      683    (8)
$D1_SBTTL        2             645    (8)     1052   (13)     1096   (14)     1158   (15)     1218   (16)     645    (8)
                                              692    (9)      775    (10)     848    (11)     919    (12)
$D2_BDATA        1             657    (8)     1061   (13)     657    (8)
$D2_BTEST        1             657    (8)     1061   (13)     657    (8)
$D2_SBTTL        1             645    (8)     1052   (13)     645    (8)
$DEF             1             82     (2)
$DEFINI          1             78     (2)     78     (2)      79     (2)      82     (2)      84     (2)      85     (2)
                                              86     (2)      88     (2)
$DS_ABORT        1             476    (3)     1291   (16)     476    (3)
$DS_BCOMPLETE    1             1290   (16)    1290   (16)
$DS_BGNMESSAGE   1             573    (7)     573    (7)      581    (7)      597    (7)      615    (7)      624    (7)
                                              631    (7)
$DS_BGNMOD       1             66     (2)     66     (2)
$DS_BGNSUB       1             742    (9)     1125   (14)     1188   (15)     1251   (16)     742    (9)      815    (10)
                                              886    (11)     982    (12)
$DS_BGNTEST      1             657    (8)     1061   (13)     657    (8)
$DS_BREAK        1             1012   (12)    1012   (12)     1048   (12)     1302   (16)
$DS_CKLOOP       1             687    (8)     1043   (12)     1149   (14)     1212   (15)     687    (8)      758    (9)
$DS_CLRVEC_S     1             1045   (12)    1045   (12)     1280   (16)     1281   (16)     1286   (16)     1287   (16)
$DS_DSADEF       5             79     (2)     79     (2)
$DS_DSDEF        2             80     (2)     80     (2)
$DS_DSSDEF       8             84     (2)     84     (2)
$DS_ENDDATA      1             657    (8)     1061   (13)     657    (8)
$DS_ENDMESSAGE   1             578    (7)     578    (7)      594    (7)      612    (7)      621    (7)      628    (7)
                                              640    (7)
$DS_ENDMOD       1             1304   (16)    1304   (16)
$DS_ENDSUB       1             771    (9)     1046   (12)     1152   (14)     1215   (15)     1300   (16)     771    (9)
                                              844    (10)     915    (11)
$DS ENDTEST      1             1048   (12)    1048   (12)     1302   (16)
```

```
$DS_ENVDEF        2             66     (2)       66     (2)
$DS_ERRDEF        1             83     (2)       83     (2)
$DS_ERRHARD_S     1             667    (8)       1003   (12)      1010   (12)      1042   (12)      1071   (13)      1083   (13)
                                                 1143   (14)      1147   (14)      1206   (15)      1210   (15)      1288   (16)
                                                 667    (8)       679    (8)       756    (9)       841    (10)      910    (11)
$DS_EXIT          1             669    (8)       1073   (13)      1075   (13)      1086   (13)      1088   (13)      669    (8)
                                                 671    (8)       682    (8)       684    (8)
$DS_HPODEF        2             85     (2)       85     (2)
$DS_KA_DEF        3             86     (2)       86     (2)
$DS_LOADPCS_S     1             1297   (16)      1297   (16)
$DS_PAGE          1             643    (7)       1050   (12)      1092   (13)      1154   (14)      643    (7)       690    (8)
                                                 773    (9)       846    (10)      917    (11)
$DS_PARDEF        1             81     (2)       81     (2)
$DS_PRINTB_S      1             576    (7)       576    (7)       582    (7)       588    (7)       598    (7)       606    (7)
                                                 620    (7)       627    (7)       632    (7)       633    (7)       634    (7)
                                                 638    (7)
$DS_PRINTF_S      1             670    (8)       1074   (13)      1087   (13)      670    (8)       683    (8)
$DS_PRINTSIG_G    1             475    (3)       475    (3)       637    (7)       639    (7)
$DS_PRINTX_S      1             590    (7)       590    (7)       608    (7)
$DS_PROBE_S       1             673    (8)       1077   (13)      673    (8)
$DS_PSLDEF        1             82     (2)       82     (2)
$DS_SBTTL         1             645    (8)       1052   (13)      1096   (14)      1158   (15)      1218   (16)      645    (8)
                                                 692    (9)       775    (10)      848    (11)      919    (12)
$DS_SCBDEF        3             78     (2)       78     (2)
$DS_SECDEF_S      1             87     (2)       87     (2)
$DS_SETVEC_S      1             984    (12)      1090   (13)      1254   (16)      984    (12)
$EQU              1             82     (2)       66     (2)       80     (2)       81     (2)       82     (2)
$EQULS1           1             82     (2)       66     (2)       80     (2)       82     (2)
$EQULST           1             66     (2)       66     (2)       80     (2)       82     (2)
$GBLINI           2             66     (2)       66     (2)       80     (2)       81     (2)       82     (2)
$OFFDEF           1             83     (2)       83     (2)
$PR750DEF         1             88     (2)       88     (2)
$PSLDEF           2             82     (2)       82     (2)
$PUSHADR          1             667    (8)       1004   (12)      1011   (12)      1042   (12)      1071   (13)      1083   (13)
                                                 1143   (14)      1147   (14)      1206   (15)      1210   (15)      1289   (16)
                                                 667    (8)       679    (8)       757    (9)       841    (10)      910    (11)
$VIELD            1             66     (2)       66     (2)       81     (2)       82     (2)
$VIELD1           1             82     (2)       66     (2)       81     (2)       82     (2)
ECKAX_SECDEF      1             87     (2)       87     (2)

                                              +------------------------+
                                              ! Performance indicators !
                                              +------------------------+

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                    47    00:00:00.12    00:00:00.60
Command processing               912    00:00:00.74    00:00:01.51
Pass 1                          1298    00:00:13.67    00:00:19.99
Symbol table sort                  0    00:00:00.58    00:00:00.58
Pass 2                           116    00:00:03.54    00:00:04.76
Symbol table output               38    00:00:00.24    00:00:00.65
Psect synopsis output              2    00:00:00.03    00:00:00.03
Cross-reference output             6    00:00:01.20    00:00:01.39
Assembler run totals            2421    00:00:20.12    00:00:29.51
```

```
The working set limit was 3212 pages.
201200 bytes (393 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 550 non-local and 83 local symbols.
1305 source lines were read in Pass 1, producing 34 object records in Pass 2.
132 pages of virtual memory were used to define 59 macros.

                                              +----------------------------+
                                              ! Macro library statistics !
                                              +----------------------------+

Macro library name                             Macros defined
------------------                             --------------
DISK$DIAGPACK02:[TAYLOR.ECKAX]ECKAX.MLB;1                 1
SYS$COMMON:[SYSLIB]DIAG.MLB;955                          49
SYS$COMMON:[SYSLIB]STARLET.MLB;2                          9
TOTALS (all libraries)                                   59

903 GETS were required to define 59 macros.

There were no errors, warnings or information messages.

MACRO/LIS=ECKAX8.LIS/CROSS ECKAX8
```